VISUAL ATTENTION FOR HIGH SPEED DRIVING

A Dissertation
Presented to
The Academic Faculty

By

Paul Drews

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2019

VISUAL ATTENTION FOR HIGH SPEED DRIVING

Approved by:

Dr. James M. Rehg
Interactive Computing
Georgia Institute of Technology

Dr. Evangelos A. Theodorou
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Byron Boots
School of Electrical Engineering
Georgia Institute of Technology

Dr. Dhruv Batra
School of Computer Science
Georgia Institute of Technology

Dr. Dieter Fox
Paul G. Allen School of Computer Science & Engineering
University of Washington

Date Approved: November 26, 2018

It’s a magical world, Hobbes, ’ol buddy... Lets go exploring!

Bill Watterson

To my wife.

ACKNOWLEDGEMENTS

Thanks to my primary collaborators throughout my PhD, Brian Goldfain and Grady Williams. Without their help, work and the synergy of our PhD topics, none of this would have been possible. Thanks to James Rehg, whose insight, guidance and support have made my PhD journey a fruitful and fulfilling one. Thanks to the many students who have worked on the AutoRally project and helped bring this platform to life. And thanks to my wife Shan for supporting me through this adventure.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Coupling of control and perception is an especially difficult problem. This thesis investigates this problem in the context of aggressive off-road driving. By jointly developing a robust 1:5 scale platform and leveraging state of the art sampling based model predictive control, the problem of aggressive driving on a closed dirt track using only monocular camera images is addressed. It is shown that a convolutional neural network can directly learn a mapping from input images to top-down cost map. This cost map can be used by a model predictive control algorithm to drive aggressively and repeatably at the limits of grip. Further, the ability to learn an end-to-end trained attentional neural network gaze strategy is developed that allows both high performance and better generalization at our task of high speed driving. This gaze model allows us to utilize simulation data to generalize from our smaller oval track to a much more complex track setting. This gaze model is compared with that of human drivers performing the same task. Using these methods, repeatable, aggressive driving at the limits of handling using monocular camera images is shown on a physical robot.

CHAPTER 1
INTRODUCTION

A largely unexplored regime that autonomous vehicles must master before their introduction to public roads is abnormal and extreme driving conditions. These "corner" cases include pre-collision regimes, maneuvering at high speed, and driving on surfaces other than asphalt and concrete such as ice, gravel, and dirt, where severe understeer, oversteer, skidding, and contact loss with the road surface is common. Additionally, dynamic, unpredictable environments such as close proximity to moving vehicles, pedestrians, and other obstacles require short time scales for perception, planning, and control that challenge traditional perception, planning and control methodologies.

In order to effectively master these extreme driving conditions, perception and control must be closely informed by each other. This close coupling is required in order to perceive the world quickly while performing aggressive maneuvers. Existing commercial solutions for driver assistance and vehicle autonomy utilize relatively simple models of vehicle dynamics, and emphasize the integration of multiple sensing modalities to characterize the vehicles environment. Several examples of this approach can be found in the perception and control architectures utilized in the DARPA Grand Challenge Competitions [1, 2, 3].

In this thesis we will concentrate on the task of *aggressive driving*. We define aggressive driving as a vehicle operating close to the limits of handling, often with high sideslip angles, such as may be required for collision avoidance or racing. Because aggressive driving leaves little room for error, it is a perfect space to showcase fast algorithms for coupled perception and control. Being at the limits of performance, especially in a track setting means we may need to know or guess about the environment before it is fully visible to our sensors.

We chose to explore this problem in the context of off road racing. In a controlled

setting such as a race course, we are able to repeatably explore the limits of handling for a vehicle. Because we explore racing in an off-road setting, aggressive sliding and drifting maneuvers are possible. Additionally, bumpy and high dynamic off-road environments introduce potentially large disturbances for the controller to reject, and large vibrations and high dynamic camera movement induced by large bumps on the track.

In order to test our abilities in these conditions, we create a platform suitable for testing state of the art algorithms in aggressive driving scenarios. We believe that scaled vehicles off an excellent combination of compute and sensor payload, robustness, and ease of maintenance in aggressive driving regimes. We design and build the AutoRally platform using a commercial, 1:5 scale RC car chassis and off-the-shelf computing and sensing. AutoRally is designed for robustness, ease of use, and reproducibility, so that a team of two people with limited knowledge of mechanical engineering, electrical engineering, and computer science can construct and then operate the testbed to collect real world autonomous driving data in whatever domain they wish to study. Additionally, we develop and experimentally validate a stochastic sampling-based model predictive control algorithm for aggressive driving. This algorithm allows real time operation at the limits of handling by optimizing over the full non-linear dynamics of the system. This control method does not require explicit gradients of the cost function, so it allows us to directly use noisy neural network outputs to control the vehicle.

The major contribution of this thesis is an approach to autonomous racing in which vehicle control is based on direct computer vision sensing, using only monocular camera images acquired from a dirt track in a rally car racing environment. We address the challenge of learning visual models which can be executed in real-time to support high-speed driving. Our framework is able to take as input a single monocular camera image and output a cost map of the area in front of the vehicle. This cost map image is fed directly into a model predictive control algorithm, with no pre-processing steps necessary. Because the cost map learned by the neural network is independent of the control task being performed,

we can use any driving data, including human data, as training data and still generalize to different tasks. Additionally, because we learn an interpretable intermediate representation and an attention mechanism to focus features used from the image, it is much easier to diagnose failure cases.

Additionally, we develop an end-to-end trained attention mechanism, which is only trained to optimize task performance. We show that this network has two desirable properties. First, it provides lower error and better generalization than other neural network architectures tested. Second, the attention learned by this network is both "reasonable" and qualitatively similar to measured human gaze on exactly the same task. By reasonable, we mean that it follows findings from previous research on human attention and salient features when driving and racing.

Our end product is the combination of these individual pieces, the AutoRally platform, MPPI controller, and cost map prediction, into a system that is able to consistently race off-road. We demonstrate this ability on two test tracks we create, achieving human level performance in both environments using only onboard images. We are able to match the performance of the test driver who provided all system identification data using only onboard monocular images. Using our attention mechanism, we are able to visually generalize from a simple oval track to a much more complex track geometry. We are able to use low fidelity simulation images to enhance this generalization performance. We thoroughly quantify the strengths and weaknesses of the different network architectures and how the full system performs.

1.1 Related Work

Because we tackle the problem of developing and testing a full autonomous race car on a physical platform, we must draw from a variety of fields of prior work. We will be drawing from a wealth of knowledge in previously built autonomous platforms ([1, 4]) to show the uniqueness of our platform. Optimal control theory is a natural framework in which frame

the problem of driving quickly, but we new innovation to make this approach tractable and robust to our estimator noise. Simultaneous Localization and Mapping (SLAM) [5, 6], semantic segmentation [7], and other computer vision approaches to driving such as [8] heavily influence our work. The fields of active vision [9] and saliency [10], along with a great deal of work on attention models [11] and general neural network structure in computer vision, guide us toward our final inference structure.

1.1.1 Scaled Platforms

Scaled platforms constructed from modified RC cars are popular in the academic and hobby communities. These platforms are typically 0.2 m to 1 m long and weigh between 1 kg and 25 kg. Costs range from a few hundred to tens of thousands of dollars, largely determined by the size, sensors, and computing. Construction, maintenance, and programming is typically handled by a small team of students or researchers. Recently, several open source projects released complete documentation and interface software, which is in contrast to the one-off nature of older work that often lacked enough information to replicate.

Documentation for open source platforms normally includes parts lists, build instructions, and interface software for the sensors and actuators. Availability of tutorials, simulation environments, and public datasets vary by project. Common sensors include wheel speed, inertial measurement unit (IMU), cameras, depth sensors, ultrasonic, and light detection and ranging (Lidar) units. The target environment for these platforms is typically indoors on a smooth surface. The Donkey Car [12] is an easy to build 1:16 scale autonomous platform for the DIY Roborace events targeted at hobbyists. Onboard computing and sensing are a Raspberry Pi 3 with a matching wide angle camera. The Berkeley Autonomous Race Car (BARC) [13] is a 1:10 scale vehicle designed as a simple and affordable research platform for self-driving vehicle technologies that has been successfully used to demonstrate various control algorithms. The onboard ODROID-XU4 is similar in computational performance to the Raspberry Pi 3, and the sensor suite in-

cludes a hobby grade camera, IMU, four ultrasonic range finders, and Hall effect wheel speed sensors. The F1/10 project [14] and accompanying Autonomous Racing Competition allows teams to race against one another using a common 1:10 scale platform developed at the University of Pennsylvania. Computing on the F1/10 platform is performed by an Nvidia Jetson. The sensor suite includes a hobby IMU, compact indoor Hokuyo 2D Lidar, and optional Structure and Zed depth and motion sensing cameras. The 1:10 scale Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot (RACECAR) [15] from Massachusetts Institute of Technology is a platform for researchers creating applications for self driving cars. RACECAR also uses the Nvidia Jetson for computing, and includes the same Hokuyo Lidar and Zed stereo camera as the F1/10 platform.

While all of these platforms are easy to build, moderately priced, and offer some onboard sensing and compute capabilities, their design limits their use to smooth surfaces, typically indoors. All of the platforms lack a global position system (GPS) device, which is a common sensor for outdoor vehicles. Instead of GPS, global position information can be provided by instrumenting the environment such as a VICON external motion capture system or beacons rigidly mounted around the environment. These systems restrict the possible operating space to a couple hundred square meters because of sensor field of view and resolution restrictions, and are priced in the tens of thousands of dollars for out-of-the-box solutions. The chassis, mounts, and enclosures of the platforms are typically not designed for repeated crashes and collisions that are inevitable when testing novel autonomous vehicle technologies, so the delicate sensors and electronics are easily damaged when something goes wrong. Onboard computing is inadequate for much of the state of the art research because of size and power limitations. This necessitates significant code optimization or offloading of computation to a remote computer. Off-board computation introduces its own set of problems including increased latency, dependence on a reliable, high bandwidth wireless connection, which dictates the size and configuration of testing environments. The limited payload capacity and power availability also severely limits the

ability to test new sensors such as a Lidar and high frame rate machine vision cameras because the size, weight, and data rates quickly overwhelm the platforms.

Many one-off experimental platforms have been created for specific projects. In [16], a model predictive control (MPC) algorithm running on a stationary desktop computer with a motion capture system has been used to drive a custom 1:10 scale RC platform around an indoor track with banked turns, jumps, and a loop-the-loop. Platforms were developed to test autonomous drifting controllers in [17] and [13], and to push scaled autonomous driving to the friction limits of the system in [18]. A framework with multi-fidelity simulation and accompanying hardware platform for use in reinforcement learning problems relating to autonomous driving was presented in [19]. A 1:5 scale autonomous platform was developed to investigate stability control in [4, 20]. While these platforms were successfully used for the experiments in their respective publications, there is not enough public information available to be able to build, operate, and program one without essentially starting from scratch.

1.1.2 Optimal Control

Despite the mathematical appeal of the problem formulation admitted by optimal control theory, it traditionally has not been utilized in the context of autonomous driving. The most popular current methods for controlling autonomous vehicles have their roots in the DARPA Grand and Urban Challenges, where the winners used a hierarchical approach that split the control problem into two sub-problems: path planning and path tracking using a feedback control law [21]. In these methods, a path satisfying some driving-related constraints is first planned, and then this path is used as the input to a low-level feedback control law which computes the steering and throttle commands to be used.

While the hierarchical approach makes the control problem tractable, and has many successful applications to autonomous vehicles [22, 2, 23, 24, 25], the decomposition into planning and execution phases introduces inherent limitations. In particular, the path plan-

ner typically has very coarse knowledge of the underlying system dynamics, usually only utilizing kinematic constraints [26, 27, 28, 29]. This means that performing maneuvers in aggressive regimes is problematic, since a planned path may not be dynamically feasible [30]. Conversely, a path planner may eliminate an aggressive yet feasible trajectory if it is limited to considering paths only in some known safe region.

Traditionally, the barrier to directly applying optimal control methods to the full autonomous driving problem has been tractability. The state space in autonomous driving is too high dimensional for global methods like solving the Hamilton-Jacobi-Bellman equation to apply, and it involves non-linear dynamics and non-convex objectives which makes applying local methods difficult. There have been a number of methods which analyzed the problem from an optimal control perspective off-line. Examples of this line of research include the work in [31] where cornering is posed as a minimum time problem and analyzed offline. In [32] an optimal open loop control sequence is computed offline, and an LQR controller is used to stabilize the vehicle about the open loop trajectory. Additionally, an approach for performing aggressive sliding maneuvers in order to avoid collisions is developed in [33], where optimal trajectories are generated offline for a variety of initial conditions and then a feedback controller is synthesized using Gaussian Process regression. However, given the complexity and sheer number of situations involved in autonomous driving it is clear that the general autonomous driving problem cannot be tackled by generating policies offline. One method which does perform simultaneous planning and tracking online with optimal control is [16], where a planner solves a boundary value problem to interpolate between way-points in order to generate a feasible trajectory for a model predictive controller. This method is capable of producing impressive acrobatic maneuvers, however, it relies on a dense series of waypoints to reduce the cost function to a quadratic optimization objective, and introducing additional non-quadratic terms or constraints would be non-trivial.

1.1.3 Attention

The study of attention in humans, specifically computational models of attention, is a very broad field. From fairly simple models of bottom up attention using center-surround differences [34] to complex, end-to-end trained models of attention [35], many models have been proposed and used to show benefits over full image processing in neural networks. Our work draws primarily on end-to-end trainable computational models. However, early work in active vision and image saliency underpin our ideas of what these end-to-end models should learn and what a "good" attention model looks like. Where active vision seeks to actively control an observer to maximize information gain, saliency seeks to determine what parts of a static image are salient. Additionally, there is a body of work on where human drivers look in a scene, both in racing contexts and in every day driving scenarios that can establish expectations for our models.

Active Vision

The active vision field in computer vision has garnered considerable attention in the past, with works from Aloimonos, Kelly, Bacjsy, and Davison trying to determine the best way to think about this problem. The generally accepted definition for this problem is the active control of the cameras (by moving a mobile base, redirecting the camera, verging a stereo pair, changing focus, etc.) This early work focused on solutions such as building a detailed model of the word and selecting locations, knowing the covariance of object tracking or pose estimation and calculating mutual information, or detailed expert domain knowledge of the problem and sensor system at hand to avoid unnecessary measurements. Early work from Aloimonos [9][36] shows an analytical approach, where equations for shape from shading, contour, and texture, among others, are shown to be better posed and an easier optimization problem to solve in an active vision context. This view of the problem does shed much insight into why active vision can be useful, but in the long run better optimization algorithms (eg slam [5][6]) and faster computation solved many of these issues for a

passive observer.

Similarly, Bacjsy's early work [37][38] takes a theoretical approach, but thinks about the problem in terms of information gain. This information gain approach has had notable success such as [39] and earlier work from Davison [40], although Davison uses uncertainty instead of information gain. However, calculating the maximum information gain often suffers from high computational complexity. Some success has been found using approximations, but this approach is generally computationally infeasible.

In contrast, early work from Kelly [41] takes a much more pragmatic approach. Given a specific problem, he carefully choses which pixels to process, how quickly to process them, and how a sensor should be setup to maximize the useful information gained from a camera setup.

An early example of active vision successfully simplifying a vision problem comes from active focus and vergence control. Vergence and focus can be used independently to limit and constrain the search problem for a stereo correspondence search. In [42], camera vergence and focus are first searched to provide boundaries of objects in a scene. This in turn constrains the stereo matching problem to a small disparity window, allowing a much more accurate and robust solution than the same stereo algorithm without vergence.

Saliency

Image Saliency studies where a human will look in a scene, or what parts of an image contain the most information. Most often, the field of image salience is concerned with where a human will look in a single scene. Generally, this field can be divided into top-down [43] and bottom-up [34] models. Bottom up models are based on very low level features changes such as color or edges. They include graph-based methods [44], spectral methods [45][46], and contrast based methods [47]. All of these methods are based solely on low level image features, ignoring task level information. Top down methods such as [43] and [48] instead use external information about the task being performed to produce

salient points for the task at hand.

The output of a bottom-up method is generally a salience map of an image, with more contrasting or salient points highlighted. Top down methods more often are designed to work on image sequences or gaze sequences, and produce one or a series of gaze points. Because of this difference, top-down, computational methods that can predict gaze from a series of images is most relevant to our case. However, while this work can provide reference for what a human will look at, our end to end models of attention are likely optimizing something different that both saliency and active vision, so will not necessarily produce the same attention areas.

Visual Attention of Drivers

Studies such as [49] and [50] suggest that human drivers fixate on very predictable targets while driving or racing. Humans also change their gaze pattern significantly based on what task they are performing. For example, if a person is driving on an interstate and sees a car stopping rapidly, most of his or her effort will be spent watching what that car is doing and very little spent analyzing what the car two lanes to the right is doing. Much previous research in this field has focused on the tangent point model [51] of driver gaze and how the upcoming road effects the gaze point [52].

Computational models tend to fall into two categories: control theoretic models assuming gaze points and computational models of the gaze fixation points. Control theoretic models such as [51] attempt to describe plausible models of the control scheme that a human might use. These are influenced by research into how humans are effected by all of the information in their visual field [53][54] and studies into where a human fixates while they are performing a driving task [49]. These computational models excel at explaining different types of errors produced by different fixation points. They can also explain how a simple gaze model, where a human fixates either a fixed distance in front of the vehicle or at the visual apex of a corner, can result in a simple and stable control law. While this is very

helpful for showing how control theoretic notions such as stability arise from gaze patterns, it has more trouble predicting where the gaze may land in a more complex scenario.

In contrast, computational models of gaze direction use image features and task features [55] or task features and assumed salient locations based on the task being performed [56]. These models are much closer in their goal to our work, in which we produce specific locations or heat maps in the image to observe.

In the work of [55], the model and comparison with a human takes place in a simulated environment. It does not attempt to directly control the vehicle, rather it attempts to predict the location of human gaze based on control actions taken by the human and a gist summary of the image. While this is an excellent method to understand how task information can inform gaze location, it still designed to mimic human gaze location given human task inputs. Rather than optimizing for task performance, they optimize for gaze location.

In [56], a model is learned that drives a simulated vehicle. Several competing task goals of following distance to a lead car, lane keeping, and maintaining a constant speed, are introduced. A model is proposed where the information needed for these competing goals becomes noisy over time, requiring fixation on task specific locations (lead car, lane markings, and speedometer). The model then selects fixation on one of these three locations to reduce noise in the corresponding estimate. While this model does produce a controller that will drive a vehicle, it makes the assumption that the correct fixation locations are known in advance, requiring experiments to be performed in a controlled simulation environment. In addition, model parameters are not tuned to maximize task performance, but rather to maximize fixation location accuracy. In our work, we chose a tangential direction, in which the model is optimized for task performance and then compared to the gaze locations produced by humans.

Autonomous racing is a very interesting problem in which to study attentional systems. There have been several studies of the gaze direction of a human driver in the context of normal road driving [49] and racing [50]. These works show that humans in both typical

driving scenarios and racing scenarios attend most closely to the inside of an upcoming corner, and to the furthest visible point. Humans have evolved a perceptual system that is highly optimized for survival, and thus is extremely efficient and accurate. These works attempt to quantify and explain the gaze strategies used by human when performing this task. Because racing drivers can consistently perform the task of high speed driving on a closed course at a higher level then their peers, it is reasonable to suppose they have optimized their gaze strategy relative to their peers for this task. Therefore, we can see their gaze strategy as approaching an optimal strategy given the constraints of their perception system.

Learned Attention

In recent years, there have been many efforts to use attention mechanisms in an end to end learning system. In contrast to active vision or salience, these models ignore data from humans and instead use backpropagation and reinforcement learning to maximize task performance. Rather than modeling what a human does, these models develop attention as a function of the structure of the learner and the task. This attention is then an emergent function of task and data rather than imposed constraint.

These models can be very roughly categorized as hard or soft attention, depending on how the attention location information is used. These models predict either a single location (or series of locations) to attend in an image in hard attention [35], or a map of useful locations to attend in soft attention [11]. While hard attention is appealing due to its similarity to human foveal eye structure and computational advantages of processing only parts of the image, it can be very difficult to learn on hard, real-world tasks. In contrast, soft attention produces a model that is easier to learn and has shown good performance in complex, real world tasks such as image captioning. However, it still requires processing of the full image uniformly, so does not give the same computational benefits of hard attention.

Hard attention is related to the field of image saliency. Both propose specific image

locations to attend that will, in some way, provide information about the image. The model of hard attention is very directly related to and derived from the human foveal vision system. Many works, such as However, the way these locations are proposed is fundamentally different. In saliency, the general trend is to propose a model and tune or fit those model parameters to human gaze data. These models then predict gaze location in a new image. This is in contrast to hard learned attention, where the gaze function is learned end to end to best perform a task.

In [57], an early end-to-end learned model for attention is proposed. This model is based on Boltzmann Machines, and produces reasonable results on the simple MNIST dataset. In there era of deep learning, [35] proposes a model based around convolutional neural networks and the long short term memory (LSTM) variant of recurrent neural networks. This model learns separate components for predicting something about the current image and predicting a next gaze location that is most useful. In doing so, they produce interpretable gaze patterns on the simple MNIST and Street View House Numbers datasets. This model was extended in [58, 59, 60] in an attempt to make it work beyond simple datasets. Additionally, most of the models are reliant on sampling based gradient estimation, know as the REINFORCE algorithm (First talked about in [61]) instead of simply using back propagation.

In contrast to hard attention, soft attention looks at the entire image, but enforces some parts of the image be weighted more heavily than others. The semantic captioning work [11] introduces this architecture, where the image goes through some convolutions and this intermediate representation is multiplied by the output of a softmax to produce an attention mechanism. This has the strong advantage of being fully differentiable, so does not need to rely on costly sampling for training. Using the softmax layer as an attention mechanism could be thought of as a form of regularization in this context. It is understood that, in general, most regions of the input image are distractors and do not correlate to correct task performance and task generalization. By multiplying by a softmax taken in the 2d spatial

dimension, this work formally enforces the notion that some spatial regions are important and other are not, giving a direct method for rejecting these unimportant regions.

1.1.4 Autonomous Racing

Several different approaches have been taken to the problem of aggressive autonomous driving, and autonomous driving in general. In [62], an analytic approach is explored. The performance limits of a vehicle are pushed using a simple model-based feedback controller and extensive pre-planning of a racing line to follow around a track. More recently, [63] showed the benefits of model predictive control on a 1:10 scale vehicle following waypoints through a challenging obstacle course. [64] also shows some of the benefits of model predictive control in an outdoor, dirt environment.

However, these approaches all rely on highly accurate position from an external source (either GPS or motion capture). When approaching autonomous driving in natural environments, we would prefer being able to drive aggressively using only internal sensors, and ideally only low-cost sensors such as cameras and IMU. There are several ways to approach this problem.

There are many simultaneous localization and mapping (SLAM) approaches that use cameras [5, 6], LIDAR [65], or other sensor combinations [66] to provide accurate position. These systems typically provide position relative to a generated map. However, this approach can be very challenging when localizing in a map created in significantly different conditions [67]. Because a large map needs to be created and position calculated relative to this map, these methods tend to be computationally expensive. An alternative method to providing absolute position uses deep neural networks to directly regress a position estimate in an area previously visited [68]. However, this method of localization is not yet sufficiently accurate to be directly used for control.

Instead of relying directly on accurate localization, one can instead rely on camera images to derive actions directly. In these works, a policy directly from camera images

In [69], a strong case is made for end to end learning, or behavior reflex control in the context of autonomous driving. In this paper, the authors are able to train a neural network to directly regress a steering control signal from images given only training data of humans driving on many different types of roadways. This work follows from seminal work by Dean Pomerleau in the Alvin project [70]. This regime can be extended to other robotics domains as well, such as [71], where a neural network is trained as a policy from images to manipulator arm torques using guided policy search.

Autonomous driving and control as an end-to-end learning problem is an active area of research. Prior work on deep learning and model predictive control includes end-to-end methods [69, 72, 73] and encoder-decoder architectures [74, 75] to perform predictive control using raw observations. In [72] an MPC controller is used as a teacher to train a Convolutional Neural Network that maps raw observations such as wheel speed, acceleration, and images directly to throttle and steering. This paper also uses the AutoRally platform and the same track setting as used in this work, so makes an excellent direct comparison to our work. While it elegantly encompasses the our entire system as a single learning problem, it has severe drawbacks. When errors occur, there is very little ability to probe the reasons for the failure. Because this solution is not modular, it must be re-learned if any part of the system changes. For example, if we change vehicles, or even change the calibration of our vehicle, and end-to-end solution my need to be completely re-learned. Because of the modularity of our system, if the vehicle changes we can simply swap out the dynamics model without needing to re-learn any other pieces of the system. End-to-end imitation learning methods are also inherently limited by their teacher. However, our system is able outperform the best lap from the training set by utilizing the raw dynamics and coming up with even better strategies.

A very close area of work comes in the form of direct affordance based control. In [76], a recurrent learning system is used to estimate liquid volumes as an affordance from video. Their goal is to autonomously pour a liquid from one container to another. While

this problem domain is vastly different from our domain of aggressive driving, the idea of learning an affordance which can be used passed to a lower level controller shares much with our architecture. However, they learn most of the required dynamics, including state propagation of the liquid, with a black box recurrent neural network. This then leaves a simple task for the controller, allowing them to use a simple PID controller. Our system has much more stringent control requirements, necessitating more accurate state estimate.

In the area of autonomous driving with affordances, [8] learns lane affordances and other vehicles directly. Again, these affordances are learned from a single monocular camera image, and are low level signals suitable for direct control. However, their control strategy is simple, and they primarily tackle simulation environments using a simple lane tracking controller. This work does not address the aggressive driving regime at all, learning affordances that are two low level for use by a complex model predictive controller such as [64]

Other solutions attempt to learn a drivability function suitable for planning or control, much as we do in our work, directly from image data. By utilizing accurate short range data provided by stereo vision, [77] learns a neural network to predict far-field traversability from images, which is then fed into a separate planning and control framework. They do utilize automatic labeling of images by projecting later traversal of an far field image region with high accuracy, short range sensors. However, this approach requires significant geometric image pre-processing, and is suitable for slow speed locomotion over unknown environments, not high speed locomotion.

Many modern autonomous vehicle solutions rely on Semantic Segmentation algorithms. This field is well studied by the computer vision community, often thanks to large scale datasets such as [78]. Lately, deep neural network architectures have achieved excellent results on semantic segmentation datasets such as [79] and [80]. These models aim to produce a per-pixel labeling of an input image. Many techniques to improve the accuracy of these models, such as conditional random fields (CRFs) [7] and dilated convolutions [81]

have advanced the state of the art in this field. This field is very active and popular given the recent interest in autonomous driving. We do not attempt to compete with these works, rather show this work as an alternative method, with tangential benefits and limitations.

High Speed Planning in Unknown Environments

Another area of research related to our problem of autonomous racing is high speed planning in unknown environments. This problem is well outlined in [82] and [83]. In this work, the primary goal is safe navigation at high speeds, using traditional planning and control methods. A simple planar lidar is used for many of these experiments, but low resolution monocular cameras are also explored. While they do achieve very high speeds and impressive movement through cluttered environments, they still rely on traditional planning methods which use holonomic vehicle dynamics are employed. This makes these methods difficult to extend to the non-holonomic, aggressive dynamics regime such as we explore.

While the suggested methods in [82] are excellent for truly unknown environments, they do not allow for driving at the limits of handling in a track setting such as ours. [83] introduces several concepts useful for the planning of high speed driving in unknown environments. They use total expected cost over over a trajectory to allow the encapsulation of probabilistic information about unknown environments. This allows safe planning in unknown spaces while still gaining the advantages of learned vision that can estimate regions not directly observable.

1.1.5 Particle Filters and Smoothing

Portions of the solution that we propose have been studied in isolation. There is a great deal of literature about localizing from camera images. One successful approach is found in the Simultaneous Localization And Mapping (SLAM) literature. Methods such [5, 6] use whole image matching and keypoints while inferring and leveraging the 3d structure of the scene. These systems typically provide position relative to a generated map and could

be suitable for our use case. However, these methods fall short in close to the ground, monocular, high dynamic and vibration video such as we have. Another approach that doesn't require explicit estimation of 3d scene geometry is to directly regress the position from camera images. Works such as [84] and more recently [68] show promise. However, they do not have the fine-grained accuracy that we require in order to drive aggressively in a tightly constrained track. Semantic segmentation methods such as [79] and [80] may be used to obtain drivability regions in the image as our method does. However, these tend to be computationally heavy and require a planar projection or depth data to transform to the cost map representation we use.

Additional closely related work shows the integration of particle filtering and vision based observations in [85]. More recent work such as [86] show more modern results other filter methods. They show good performance with the addition of lane markings, but do not utilize learning to help with the vision portion of the problem and do not address the particular problems of the aggressive driving regime.

CHAPTER 2

AUTORALLY AND MPPI

Figure 2.1: 1:5 scale AutoRally platform. Cameras, GPS, IMU, and GPU/CPU are all carried onboard

In order to explore the limits of perception algorithms during aggressive driving, a test vehicle is needed. A successful platform for this research must meed several requirements. First, it must be capable of operation at high speeds, performing aggressive off-road maneuvers, without fear of damage. Since the vehicle will be pushing its capabilities to the edge, it must be able to survive mistakes that cause crashes with a minimum of damage. Second, it must have sensors, software, and computation capabilities to support these research goals. This means a modern, high power computer and GPU, suite of light weight sensors, and time synchronization for all sensors. Third, the platform will require a baseline set of algorithms such as accurate localization and aggressive control capabilities.

Because no platform exists that meets these requirements, we chose to build our own based on a 1:5 scale radio controlled (RC) vehicle chassis. Despite recent progress and many publications detailing scaled autonomous testbeds [12, 13, 14, 15, 16, 17, 18, 19, 4, 20], much of the available results lack reproducibility because of the one-off nature of these testbeds, restrictions imposed by the use of private datasets, and inconsistent testing

methods.

To this end, the AutoRally platform was created, in collaboration with fellow researchers, and covered in detail in [87]. The platform is shown in Fig. 2.1. This robot is based on a 1:5 scale RC chassis modified for autonomous operation. The RC platform is designed to take a great deal of abuse at the hands of hobbyists, so it withstands testing and crashing well. Full size vehicles are not robust to crashes and roll overs, and require a great deal of infrastructure to test and maintain. The 1:5 scale vehicle size is an excellent compromise, with enough payload capacity for interesting compute and sensor capabilities while being small enough to pose minimal risk and be maintained and operated easily.

While the chassis is purchased off the shelf as a robust and complete unit, additional computation and sensors must be protected as well. Major onboard computation is protected within a custom aluminum enclosure designed for EMI shielding and impact and roll over protection. The stock lexan body is utilized for additional component protection. Complete construction and configuration instructions for the AutoRally platform are publicly available, and include all required computer-aided design (CAD) files for custom part fabrication, a complete parts list, and wiring diagrams [88]. This platform is robust and has fast onboard computing to allow testing of state-of-the-art algorithms.

Additionally, testing locations are needed in order to push the capabilities of this platform. In order to be able to repeatably test the limits of performance of the vehicle, we setup two dirt race tracks. These tracks vary greatly in complexity, from a simple oval to a complex, twisting track with various radius corners and complex S and hairpin turns. These tracks can be seen in Fig. 2.2. The variety allows exploring difficult recognition and generalization problems and scaling algorithms from a simple setting to a much more complex and demanding setting, with speeds of 13 m/s possible at the larger of the two tracks.

In order to test perception at the limits of handling, the ability to plan a path and control the vehicle to this plan must be developed. The Model Predictive Path Integral controller [89, 64, 90, 91], and its variants, are developed and utilized in order to meet this objective.

While this controller is not the focus of this dissertation and is not primarily developed by this author, an understanding of its operation and limitations is fundamental to understanding this work. To that end, a summary explanation of the algorithm and operation will be presented and the reader is directed to works such as [91] for a full treatment of this work.

2.1 Platform and Tracks

2.1.1 Sensors

A Lord Microstrain 3DM-GX4-25 IMU provides raw acceleration and angular rate data at 200 Hz (max 1 kHz) and fused orientation estimates at 200 Hz (max 500 Hz). A Hemisphere P307 GPS receiver provides absolute position at 20 Hz, accurate to approximately 2 cm under ideal conditions with real time kinematic (RTK) corrections from a GPS base station. Two machine vision cameras mounted on top of the compute box are Point Grey Flea3 FL3-U3-13E4C-C color cameras with a global shutter that run up to 60 Hz. Lenses are 70 degree field of view (FOV), 4.5 mm fixed focal length which provide a good trade-off between wide field of view for stereo processing and accurate stereo depth estimation. Each camera connects to the motherboard with a USB3.0 cable and is externally triggered by an Arduino Micro microcontroller with the general purpose input/output (GPIO) connector. Both cameras are connected to the same trigger signal which runs at a configurable rate, and can be synchronized to the pulse per second output of the GPS. The IMU is also connected to the pulse per second output of the GPS, allowing very high accuracy time synchronization of stereo images with IMU measurements.

To sense wheel speeds, a Hall-effect sensor and magnets arranged in a circular pattern to trigger the sensor were installed on each wheel hub. Hardware timers in the Arduino Due in the electronics box are used to accurately measure the time between magnets. Inter-magnet timing information is translated to rotation rates and sent to the compute box at 70 Hz. Inside the electronics box, the RC signals from the receiver are read by the Arduino Due at 50 Hz and sent to the compute box so that, even under manual control, the control signals

Figure 2.2: Testing sites. Top Left: Marietta oval dirt test track where most testing was performed. Top Right: Onboard Camera view of Marietta track. Middle: New CCRF track with more interesting features. Bottom: Onboard camera view of CCRF track

sent to the actuators can be recorded. This is especially useful for collecting training data where human control signals are required. The Due also receives diagnostic information from the ESC that is forwarded to the compute box.

2.1.2 Computing

A modular, reconfigurable onboard computing solution was designed that uses standard consumer computer components based on the Mini-ITX form factor. Computing hardware development outpaces advancements in almost all other components so the standard form factor, mounting method, and data connections enables the reconfiguration of sensing and computing payloads without mechanical modifications as requirements evolve. The latest iteration of the compute box houses a current generation Intel 4-core desktop processor and NVIDIA GTX1050 GPU. WiFi is used to remotely monitor high bandwidth, non-time critical data from the platform such as images and diagnostic information. A 900 MHz XBee Pro provides a low-latency, low-bandwidth wireless communication channel. The GPS on the robot receives RTK corrections from the GPS base station, transmitted over the XBee radio, at about 2 Hz to improve GPS performance. The XBee radio onboard the robot also receives a global software run-stop signal at 5 Hz, and the position and velocity of other AutoRally robots within communication range at up to 10 Hz.

2.1.3 Testing Tracks

In order to fully test this system, we built two different dirt test tracks, shown in Figure 2.2. The Marietta track is a smaller dirt oval. We have been testing at this track for several years and have a great deal of historical data from this track. The Cobb County Research Facility (CCRF) track is newer. It is about 3 times as large as the Marietta track, with more varied and challenging turns, a much faster straight section, and additional width. Both tracks have a similar dirt surface, allowing our dynamics models to work on both tracks.

2.1.4 Online State Estimation

Accurate state estimation is required for many aspects of this system. Many control methods assume accurate knowledge of the state of the system. In general, the more accurate and high-rate this information is, the better. Some states, such as the wheel speeds, can be directly measured. However, position, orientation, and velocity estimation are needed for control and planning for the AutoRally platform. Additionally, high accuracy state estimation will be used for cost map learning. Linear velocities, orientation, and sensor biases, in general, cannot be measured with the onboard sensors and some form of sensor fusion is needed to estimate these values.

GPS is inherently low rate and lacks orientation information. IMU measurements are relatively high rate but do not directly provide orientation or linear velocity information. By combining the time-synchronized signals from these two sensors, a very accurate and high rate estimate of position, velocity, and orientation can be obtained. This state information is sufficient for many advanced control systems.

Factor graphs combined with advanced inference algorithms such as incremental smoothing and mapping 2 (iSAM2) [92] allow smoothing over many types of measurements, while retaining the ability to re-linearize previous information. This reduces many of the problems found in the Kalman filter with states or measurements that are not approximately linear in the measurement time frame.

The factor graph representation of sensor fusion is a method of visualizing states and measurements as a bipartite graph (an example is shown in Figure 2.3). The factor graph has two types of nodes, *factor nodes* $f_i \in F$ and *variable nodes* $\theta_j \in \Theta$. *Edges* e_{ij} always connect factor and variable nodes. Variable nodes correspond to the unmeasured quantities to be estimated. Factor nodes correspond to probabilistic information gained from a measurement z_i about a set of variables (connected to the factor by edges). The factor graph as a whole represents the probability distribution generated by the probabilistic

information encoded in the factors

$$p(\theta_1, \theta_2, \ldots \theta_n | z_1, z_2, \ldots, z_k), \tag{2.1}$$

where θ_i is a variable that is not directly observed, and z_j is a measured variable. This function can then be factorized

$$f(\Theta) = \prod_i f_i(\Theta_i), \tag{2.2}$$

where Θ is the set of all variables in the graph and Θ_i is the set of variables connected to factor f_i by an edge. $f_i(\Theta_i)$ takes, for example, the following form for the first IMU factor in Figure 2.3 is

$$p(X_1, V_1, B_1, X_2, V_2 | \omega_x, \omega_y, \omega_z, a_x, a_y, a_z), \tag{2.3}$$

where X_i, V_i, B_i are the position, velocity, and bias state variables and ω_n and a_n are measured angular velocity and linear acceleration from an IMU.

Independence relationships in the measurements in $f(\Theta)$ are encoded in the edges e_{ij}, where each factor f_i is a function of variables Θ_j. The goal in sensor fusion is to find the variable assignment Θ^* that minimized the function $f(\Theta)$ in (2.2)

$$\Theta^* = \arg\max_{\Theta} f(\Theta). \tag{2.4}$$

Each $f(\Theta)$ can be written in terms of a difference between the measured value z_i and the predicted value from the measurement function $h_i(\Theta_i)$. By framing this in terms of log-likelihood, this maximization problem becomes

$$\arg\max_{\Theta} f(\Theta) = \arg\min_{\Theta}(-\log f(\Theta)) = \arg\min_{\Theta} \frac{1}{2} \sum_i ||h_i(\Theta_i) - z_i||^2_{\sum_i}, \tag{2.5}$$

where $h_i(\Theta_i)$ are the measurement functions relating a set of variables Θ_i to a sensor mea-

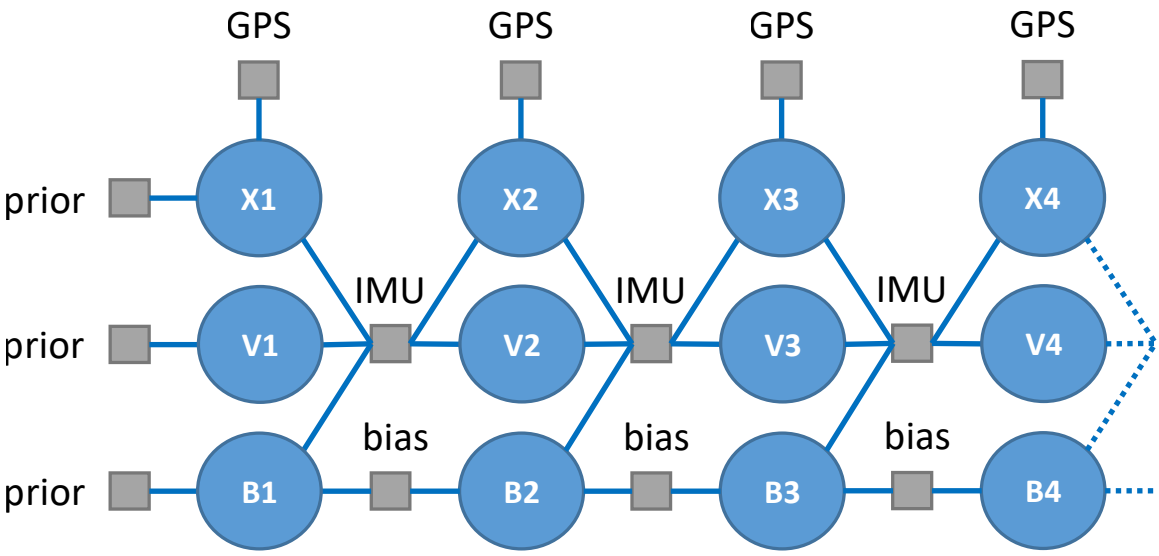


Figure 2.3: Factor graph structure used for global positioning system (GPS) and inertial measurement unit (IMU) sensor fusion using the Georgia Tech smoothing and mapping optimization library. Circles represent states and squares represent factors.

surement z_i. This minimization problem can be solved with several different nonlinear minimization strategies including Gauss-Newton or Levenberg-Marquardt that iteratively linearize and solve this problem. Using these methods, one can create an entire graph of measurements as in Figure 2.3, solve for Θ^*, and this will be the maximum likelihood estimate of the variables Θ being estimated.

However, in the case of estimating the state for the AutoRally platform, we wish to solve this problem at each time step to produce a maximum likelihood estimate of the current state variables position X_i, velocity V_i and accelerometer and gyroscope bias B_i. Re-optimizing the entire factor graph would be very inefficient, so we instead use the iSAM2 algorithm.

The iSAM2 algorithm is a part of the Georgia Tech smoothing and mapping (GTSAM) [93, 92] software package, which uses a factor graph representation to solve the smoothing and mapping problem iteratively. iSAM2 efficiently performs iterative updates to a factor graph and optimizes this new graph without re-linearizing the full problem. To perform this optimization, a factor graph, shown in Figure 2.3 is constructed with succes-

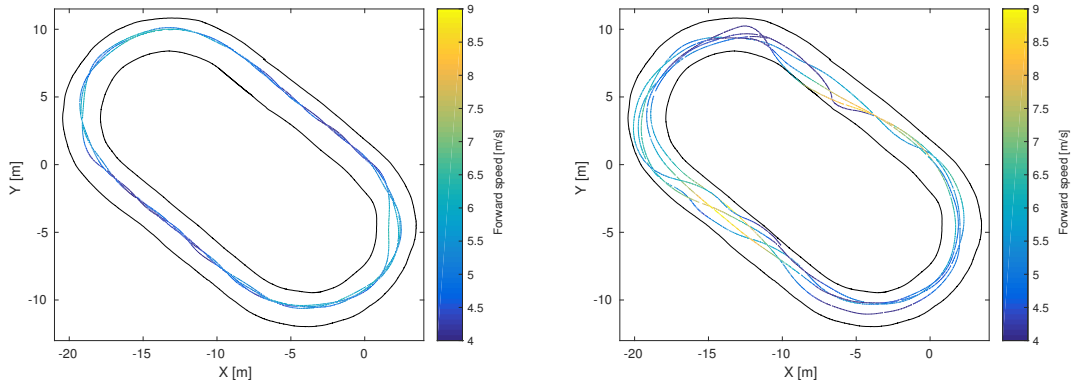


Figure 2.4: Example states generated by the state estimator built with the Georgia Tech smoothing and mapping optimization library. Inputs to the state estimator are global positioning system and inertial measurement unit sensor data. Each experiment is composed of four laps of data collected at the Georgia Tech Autonomous Racing facility oval track. Track boundaries are colored black state estimates are color coded according to the speed at which the AutoRally robot was traveling. (a) Model predictive path integral controller driving with a target speed of 6 m/s. (b) Manual driving for the system identification dataset.

sive measurements and is iteratively optimized. At each smoothing time step, an additional set of states X, V, and B are added to the graph. Additionally, measurement factors for the GPS and IMU sensors are added along with a bias smoothness factor. To keep the computational load low while maintaining high accuracy, the factor graph contains state nodes for measurements taken at 10 Hz. The factors in the graph correspond to GPS measurements and pre-integrated IMU measurements [94]. Online, the IMU measurements are integrated to interpolate the 10 Hz smoothed position to a 200 Hz state estimate with latencies on the order of milliseconds. Example trajectories are shown in Figure 2.4.

In practice, the measurements from the GPS sensor can drift slightly from day to day primarily due to the RTK correction antenna position not being fixed from one test to the next. To counteract these changes, the robot is always positioned at the same place on the track when the state estimator is started. This track position is used as the origin of a local Euclidean coordinate system, oriented tangent to the GPS reference ellipsoid. This prevents GPS drift from effecting the vehicle state estimate relative to the fixed track boundaries.

2.1.5 MPPI

Model Predictive Path Integral Control (MPPI) is a stochastic model predictive control (MPC) method designed to work with non-linear dynamics, and non-convex cost objectives. This ability to work with noisy, non-convex objective functions with no analytic derivative is the primary advantage to using MPPI in combination with learned vision. In the Direct Driving aggressive regime, presented in Chapter 3, we apply model predictive control directly to noisy cost maps produced by a neural network. Traditional planning methods can operate on cost maps such as these, but these methods are not capable of producing dynamically feasible behaviors at the edge of the dynamics of a system. Newer methods such as Rapidly expanding Random Trees (RRT) and its variants can plan a path respecting the dynamics of the system. However, these methods typically take too long to re-plan to be used directly for control, instead requiring some lower level control for trajectory tracking. It has been shown to work well in practice applied to AutoRally platform up to and beyond the friction limits of the vehicle [90].

MPPI works by quickly sampling and evaluating thousands of control sequences, and then computes the control input as a cost weighted average over the sampled controls. In order to evaluate a control sequences, a dynamics model is propagated forward in state space using the system dynamics, and each trajectory is evaluated according to a cost function. As in [90], we use a neural network model to learn the dynamics. Real time execution of MPPI on AutoRally is enabled by the onboard Nvidia GPU.

We only use a running cost function in this work (no terminal cost). The running cost that we use for generating driving behaviors from MPPI has the form:

$$q(x) = w \cdot \left(C_M(p_x, p_y), h(v_x, v_x^d), 0.9^t I, \left(\frac{v_y}{v_x}\right)^2 \right), \tag{2.6}$$

In these equations w is a vector of weights.

The first cost term, $C_M(p_x, p_y)$, is the positional cost of being at the position (p_x, p_y).

This positional cost is obtained from the cost map when the map is in use, and directly from the output of the neural network in the case of direct driving. The second term is a cost for achieving a desired speed v_x^d, the function h denotes the metric used in the cost computation. There are two different modes of driving that we use in our experiments. The first is slow/medium speed driving where the speed target actually describes the speed we want the vehicle to achieve, in this case h is the squared difference. The second mode is high speed driving, where the speed target is set to a value above what is physically possible for the vehicle to obtain (25 m/s in our case). In the second mode we use the absolute value instead of the squared difference. This was done because the absolute value has a constant gradient magnitude (wherever it exists), which enables the target speed to be set arbitrarily high without creating an exploding gradient problem. Note that even though MPPI is a gradient free algorithm, it is still sensitive to gradient magnitudes since trajectories are weighted relative to one another. The third term in the cost is a time-decaying indicator variable which is turned on if the track-cost, roll angle, or heading velocity are too high. The final term in the cost is a penalty on the slip angle of the vehicle. The full algorithm is shown in Algorithm 1 and iteratively improves an initial sampling trajectory toward the optimal trajectory under this cost function and vehicle dynamics.

Algorithm 1: MPPI

Given: $\mathbf{F}$: Transition Model;
K: Number of samples;
T: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, ...\mathbf{u}_{T-1})$: Initial control sequence;
Σ, ϕ, q, λ: Control hyper-parameters;
while *task not completed* **do**
  $\mathbf{x}_0 \leftarrow$ GetStateEstimate();
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} \leftarrow \mathbf{x}_0$;
    Sample $\mathcal{E}^k = \{\epsilon_0^k, \epsilon_1^k, \dots \epsilon_{T-1}^k\}$;
    for $t \leftarrow 1$ **to** T **do**
      $\mathbf{x}_t \leftarrow \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1} + \epsilon_{t-1}^k)$;
      $S(\mathcal{E}^k)$ += $\mathbf{q}(\mathbf{x}_t) + \lambda \mathbf{u}_{t-1}^{\mathrm{T}} \Sigma^{-1} \epsilon_{t-1}^k$;
    $S(\mathcal{E}^k)$ += $\phi(\mathbf{x}_T)$;
  $\beta \leftarrow \min_k[S(\mathcal{E}^k)]$;
  $\eta \leftarrow \sum_{k=0}^{K-1} \exp\left(-\frac{1}{\lambda}(S(\mathcal{E}^k) - \beta)\right)$;
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $w(\mathcal{E}^k) \leftarrow \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}(S(\mathcal{E}^k) - \beta)\right)$;
  for $t \leftarrow 0$ **to** $T-1$ **do**
    $\mathbf{u}_t$ += $\sum_{k=1}^{K} w(\mathcal{E}^k)\epsilon_t^k$;
  SendToActuators($\mathbf{u}_0$);
  for $t \leftarrow 1$ **to** $T-1$ **do**
    $\mathbf{u}_{t-1} \leftarrow \mathbf{u}_t$;
  $\mathbf{u}_{T-1} \leftarrow$ Intialize($\mathbf{u}_{T-1}$);

CHAPTER 3
DRIVING WITH COST MAPS

In order to extend the Model Predictive Path Integral control method beyond relying on high accuracy position from RTK GPS and IMU integration, we use monocular images from a forward facing camera. Because MPPI has proven high performance in this domain and a well tuned implementation, it makes a logical starting point. Additionally, because MPPI does not require explicit cost function gradients, it can operate directly on the noisy output of a neural network, allowing close integration between the learned neural network and the control optimization. We propose a direct-driving method where MPPI plans directly in a cost map regressed from monocular input images. This method requires no post-processing of the neural network output or other map or environment fusion. In addition to direct driving, we demonstrate performance near the vehicle limits in map-based driving by using a particle filter to integrate information from onboard IMU and wheel speed sensors with these cost map predictions.

Model predictive control works by estimating a control sequence that is optimal under a cost function and system dynamics. This cost function can be thought of as the task description, and the weights of various pieces of the cost function as the relative importance of different parts of the task. As shown before, we define our task of racing around a closed track with the following cost function:

$$q(x) = w \cdot \left(C_M(p_x, p_y), h(v_x, v_x^d), 0.9^t I, \left(\frac{v_y}{v_x}\right)^2 \right), \tag{3.1}$$

One portion of this function, $C_M(p_x, p_y)$, describes the cost of being in a position in the world, similar in concept to an occupancy grid. This is the portion of the cost that directly relies on high accuracy position and a prior cost map to compute. We note that, because

MPPI is run in a receding horizon fashion, we only need the portion of the cost function directly in front of the vehicle. Instead of using accurate pose estimation and a pre-surveyed cost map, our framework is able to take as input a single monocular camera image and output an egocentric cost map of the area in front of the vehicle that is independent of knowledge of the vehicle's position or the global structure of the cost map. This local cost map image is fed directly into MPPI which is able to optimize a control plan directly in ego-centric coordinates using this map and vehicle centric velocities.

Because the cost map learned by the neural network is independent of the control task being performed, we can use any driving data, including human data, as training data and still generalize to different tasks. This is in contrast to many reinforcement learning algorithms which must collect data on-policy. Additionally, because we learn an interpretable intermediate representation, it is much easier to diagnose failure cases.

3.1 Approach

Here we show the combination of a high performance model predictive control system with deep Convolutional Neural Networks for real-time scene understanding. MPC is well understood, and it has shown excellent results in aggressive control regimes. In [95] and our work [90], finite horizon model predictive control is used to control a scale vehicle performing aggressive maneuvers. However, these methods all rely on high quality pose estimation combined with a prior map. This works well in laboratory settings, or in highly controlled field settings. However, even in field setting using high accuracy RTK GPS, we find that the vibrations and high dynamics induced by high speed aggressive driving on a bumpy surface can prevent accurate positioning.

Traditionally, in order to drive using cameras, free space is detected in the image plane by classifying image pixels. These pixels can then be reprojected onto the ground plane or other 3d geometry. However, this is fundamentally limited to pixels that are currently visible. As the vehicle begins to travel more quickly, resolving road details and the direction

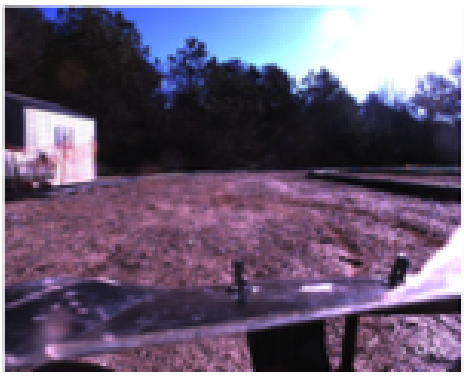


Figure 3.1: Example of onboard image with difficult upcoming corner for only segmentation.

of travel using geometric reprojection using image pixels becomes more difficult. In the domain of racing on a closed track, this problem becomes extreme. As can be seen in Fig. 3.1, the perspective from our vehicle makes driving via segmentation and reprojection alone impossible. At 13 m/s, the maximum speeds our vehicle attains, resolving track features just one second in front of the vehicle requires at least 13m of range. However, with our moderate angular field of view of 70 degrees, track pixels 1m apart in horizontal distance on the ground plane are only separated by 2 image pixels. This problem is illustrated in Fig. 3.2. Here you can see both the limited field of view of a semantic segmentation approach, as well as distant pixels being smeared across the ground plane due to the reprojection from a low camera height. This means that by the time a corner is easily detectable using a semantic segmentation approach, it is much too close to stop for in the high speed case.

Additionally, the direction of the track further through the corner is not directly observable. This limitation is caused by the field of view in our cameras, chosen to enable stereo disparity processing as well as scene understanding. However, this same limitation occurs due to occlusion in other contexts, so is a more general problem. In addition to making planning from direct drivable region estimation in the image plane ill-posed, obstacles may occlude regions of the ground plane that are within the camera field of view.

Instead, we show that using a neural network to directly predict a top-down view of

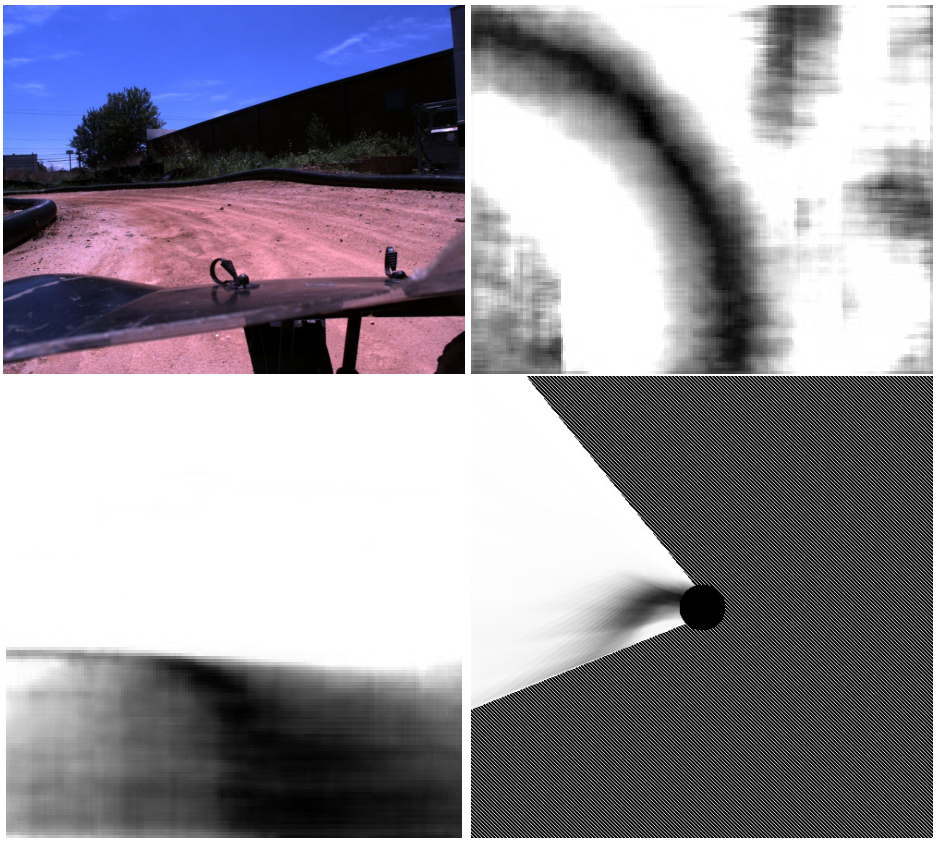


Figure 3.2: Top down and image plane comparison. Top: Input camera image and top-down costmap Bottom: Image Plane neural network output and reprojection to ground plane (White cone is camera field of view). This illustrates information loss when projecting, and how the top down network is able to use other visual cues to correctly output further track regions.

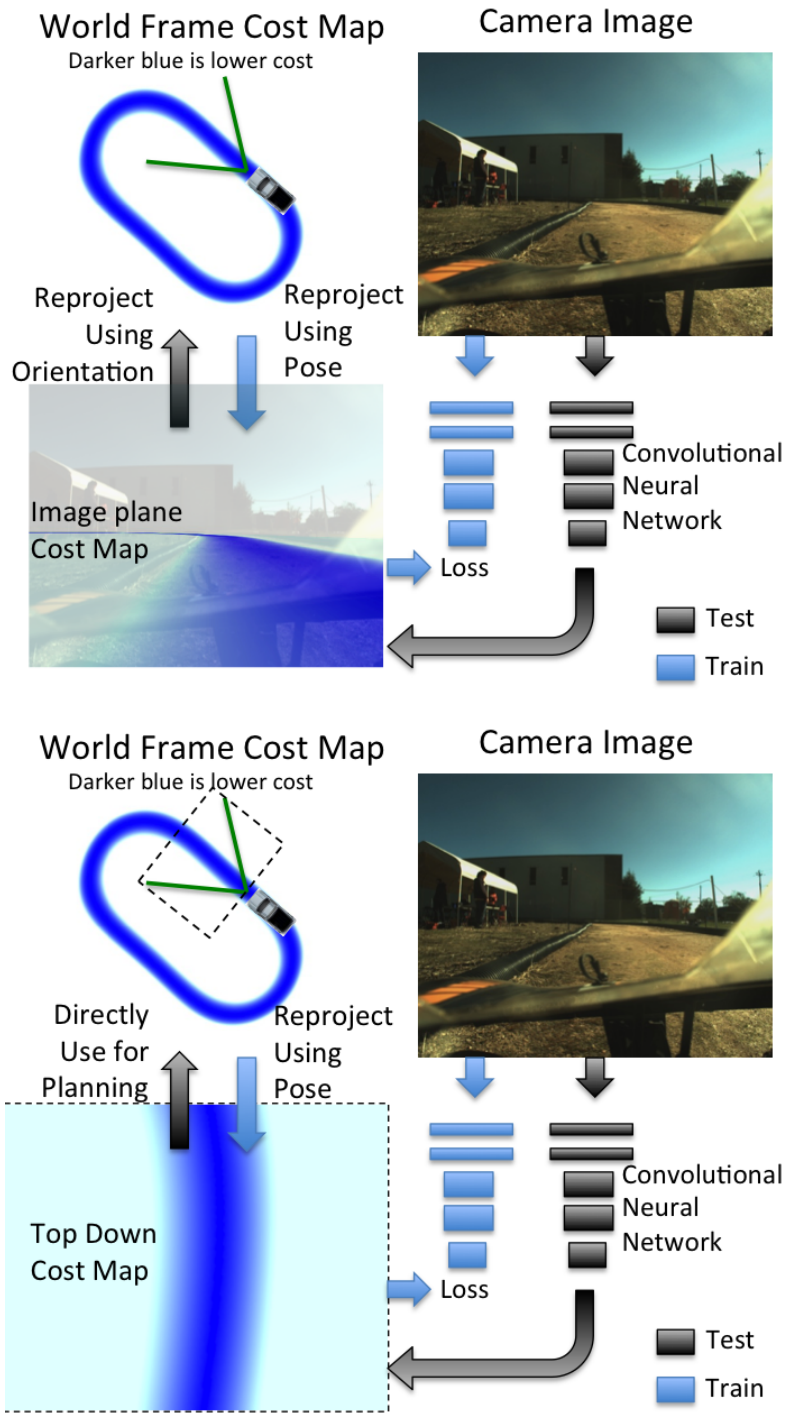


Figure 3.3: (top) Image plane cost map regression. Camera image and position on a world map are combined to label track pixels of images. (bottom) A top down projection of the cost map used as a training target.

drivable regions produces much better overall system performance. The neural network is able to use whole image clues and extrapolate drivable regions in such a way that the end result is useable directly for driving via model predictive control without any post processing. We refer to this approach as Direct Driving. The training and testing pipelines for these approaches, both image plane and top down, are shown in Fig. 3.3. The poses collected while the vehicle is running on the track and a pre-surveyed cost map are combined to produce a target egocentric cost map. This is combined with the input image collected from the onboard camera to produce an input-output training pair. These are used off-line to train a neural network. During test time, a new image is fed into the neural network to produce a cost map. This cost map is used by MPPI to plan a control sequence and execute the first action in this sequence. This process is then repeated with a new image, sequence and current action.

By factoring the control and perception tasks, we can take advantage of the strengths and mitigate the weaknesses of both deep learning for vision and model predictive control. This factoring makes the perception task of mapping images to cost functions is independent of the control policy, which means that data can be collected from many different (off-policy) sources. This mitigates the main difficulty in deep learning, which is collecting large amounts of data. In the case of model predictive control, we are able to operate without a pre-surveyed cost function and localization, enabling its usage in novel environments. However, we are still able to utilize MPC for the difficult problem of online optimization with non-linear dynamics and costs. This allows model predictive control to optimize the dynamics which is mathematically well understood, while utilizing the power of deep learning for image understanding which lacks a rigorous mathematical theory.

The experiments in this chapter compare semantic segmentation style driving, where free space is detected in the image plane, with our Direct Driving method directly regressing a top down cost map. In our implementation, the cost function takes the form of an occupancy-grid style cost map, as shown in Figure 4.3. The network is trained so that

the cost is lowest at the center of the track, and higher further from the center, in a distance from centerline style representation. Our CNN architecture is constrained to run in real time on the low power Nvidia GTX750Ti available on our platform, and it produces a dense costmap output. We found that a fully convolutional network that outputs a dense costmap with large input receptive fields produces the most accurate result. We trained this architecture to output two different types of predictions (as shown in Figure 3.3), these are (1) a top-down cost map that can be used directly by MPPI, and (2) an image-plane labeling of pixels that must be projected onto the ground before use.

Additionally, to extend the cost map prediction and further push the speed and aggressiveness of the vehicle, we develop a particle filter estimator for the vehicle state which treats the generated cost maps as a sequence of measurements. This is in contrast to directly controlling the vehicle from the local cost maps. While this removes the ability to directly drive from the neural network output, we only require a schematic map of the area that we are driving in. Most slam and localization approaches require a full reconstruction or sparse point reconstruction.

Our architecture incorporates two dynamic measurement processes: an LSTM based neural network that processes on-board video to estimate the evolving cost map, and a particle filter state estimator which drives the model predictive control algorithm. A benefit of this approach is that the vision-derived cost maps can be combined in a principled way with other sensors, such as an IMU and wheel speed encoders. This approach also naturally decouples the state estimator and controller, which can leverage mature, well-understood technologies such as particle filters and MPC, from the video-based deep neural network model which provides ”black box” estimates of the cost map. Thus the vehicle state provides an interpretable and useful latent representation which is helpful in diagnosing issues with the cost map predictor. This is in contrast to standard end-to-end approaches to learning control [72], which typically lack such informative intermediate latent representations.

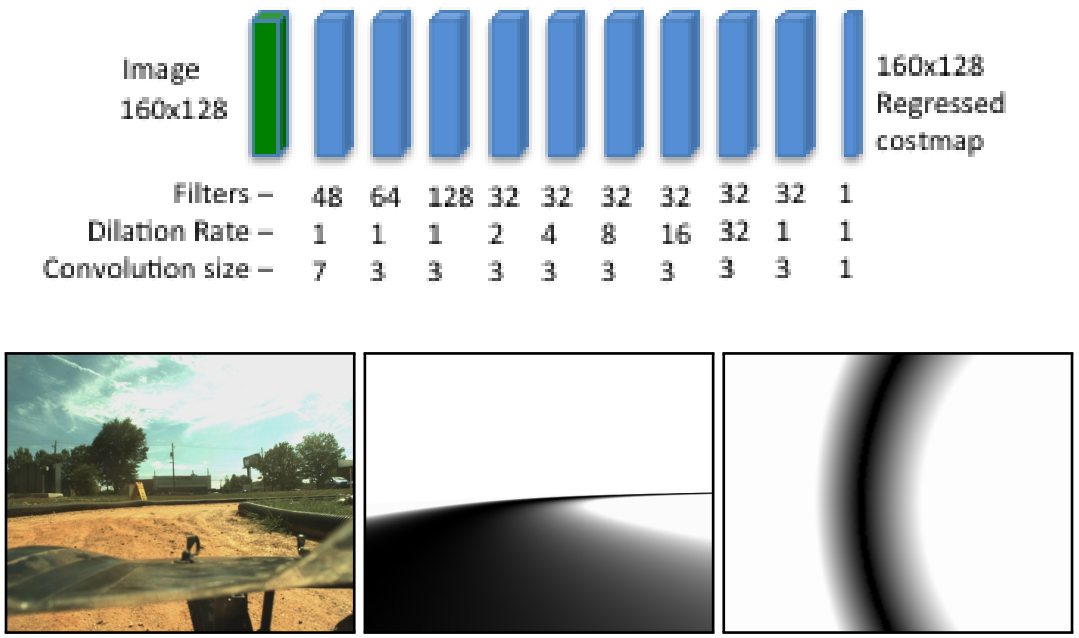


Figure 3.4: Network architecture with input and training targets. Left: Neural network architecture used to produce top down cost maps. Right: example input image, image plane training target and top down training target, respectively

3.2 Ground truth generation

In order to learn a pixel-wise regression function capable of producing traversal costs at every pixel, training data is needed on the order of 100s of thousands of frames. Labeling all of this data by hand is laborious, slow, and prone to errors. However, the AutoRally vehicle that we use in our experiments is equipped with a high accuracy RTK GPS system and IMU. These measurements are smoothed in real time to create a very high accuracy position and orientation solution, as described in Sec. 2.1.4 . Combined with a surveyed track map registered to GPS coordinates, these can be used to create hundreds of thousands of labeled images without any manual labeling of individual images. In many large-scale commercial vehicle applications, a great deal of effort is put into creating and localizing in high accuracy, large scale environment maps. As the vehicles drive in the environment, it is perfectly reasonable for them to be able to localize in a prior, high quality map in order to produce ground-truth drivability images.

In the image plane case, a homography is needed in order to transform a surveyed map on the flat ground plane to its projection in the image plane. By calibrating the transformation between the IMU (where position and orientation estimates are centered) and the camera, and calibrating lens distortion in the camera lens, a homography matrix can be computed that transforms the surveyed track map from world coordinates to image plane coordinates. This homography matrix is

$$H = kT_{im}^{car}T_{car}^{world} \tag{3.2}$$

where T_{car}^{world} is the position of the car in world coordinates (estimated at the IMU), T_{im}^{car} is the transformation between the IMU and camera reference frames, k is the camera intrinsics matrix, and H is the 3x3 homography matrix. Given this, points in the image plane can be projected to their corresponding ground coordinate frame points using

$$p_{im} = \hat{H}p_{world} \tag{3.3}$$

$$\hat{H} = \begin{bmatrix} H_{11} & H_{12} & H_{14} \\ H_{21} & H_{22} & H_{24} \\ H_{31} & H_{32} & H_{34} \end{bmatrix} \tag{3.4}$$

where p_{im} and is a homogenous image point and p_{world} is a 3D world coordinate. Using this scheme, ground truth images can be produced for each image in our training set. This mapping in not perfect due to small errors in time synchronization and violations of the assumption of constant height above ground of the camera. However, despite these small errors, the reprojected cost maps are very good, and networks are able to learn from them. To produce ground truth images for the top down network, a 160x128 section of the cost map directly in front of the vehicle(in vehicle centric coordinates) is used.

In order to train the top down method, we adopt a centerline representation. For each

track, we survey the center of the track, and store this as a set of equally spaced points. To generate an egocentric cost map associated with an image and location, we first find the centerline point closest to the location. This is used to cull track points that far away on the track surface, such as an adjacent stretch of track. Requiring the neural network to learn these features only leads to overfitting. We then scale the points to the size image we wish to produce and perform a distance transform to achieve images as seen in the bottom left of Fig. 3.3.

Using this method, we created two datasets, one for the Marietta track and one for the CCRF track, with approximately 400,000 total images with corresponding ground truth poses and surveyed cost maps. These training images were taken from several years of operation and span a wide variety of conditions. It includes substantial variability in lighting conditions, people and equipment present at the collection site, and poses of the camera on the track. This data was collected during a variety of different testing events under different types of control, including autonomous and manual driving, high and slow speed driving. For neural network training, it is split into validation and train sets.

Figure 3.5 shows an example of the raw accuracy of a neural network as a function of the position of the vehicle on the track. As a more intuitive way to understand the error, in the rest of this paper we report the average per-pixel frame accuracy as

$$A_t = 1 - \frac{1}{N} \sum_{(u,v)} |I_{u,v} - \widehat{I}_{u,v}| \tag{3.5}$$

where N is the number of pixels per image and $I_{u,v}$ is a normalized pixel value in the range $0 - 1$. For accuracy over sequences, each neural network computes an output per frame, and final accuracy reported as the mean per-frame accuracy.

In order to truly test the performance of a neural network designed for autonomous driving, it must be implemented and tested on a physical platform. While testing on datasets can show a great deal about how a neural network might perform and in what cases it may

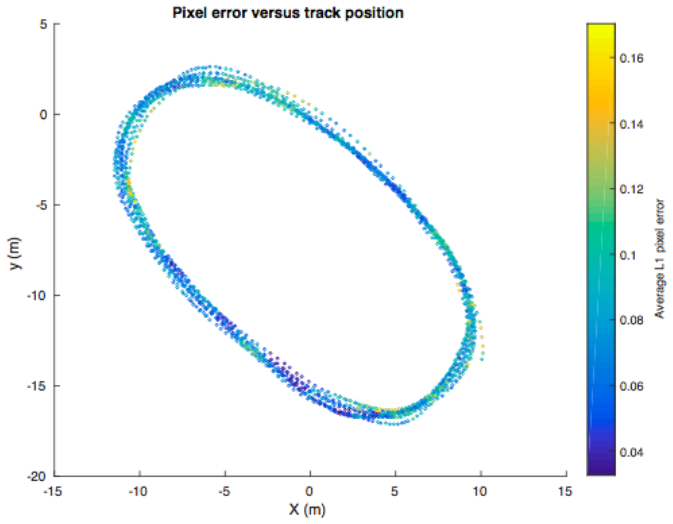


Figure 3.5: Neural network average error, as L1 distance (0-1 full scale). It is apparent from these graphs that the network has more difficulty producing accurate cost maps in corners, potentially due to a lack of context.

fail, there is no substitute for real-world testing on a robot. In our case, we test on the AutoRally platform (see Figure 2.1) described in 2. It includes all sensors and computation required onboard the vehicle. During testing, all computation is performed on the vehicle in real time, including neural network forward inference and model predictive control computation. The primary functions of the testing setup are:

- Image capture using onboard Point Grey cameras
- Deep model forward inference in Tensorflow to produce a cost map from an image
- Generation of homography using IMU based orientation and velocity
- Trajectory cost lookup and control computation
- Forward propagation of controller state, so a single image can continue to be used by the controller until a new image is available from the deep model.

All software is written using the Robot Operating System (ROS) framework and custom software packages to communicate with the AutoRally system. Forward inference through

the neural network is handled asynchronously, and the cost maps are fed to the MPPI control algorithm at approximately 10Hz for the image plane network and 40Hz for the top down network, and the MPPI controller runs at 40Hz. Velocity and acceleration information is obtained from the onboard GPS-IMU system, but absolute position is not used. For the top down case, the camera orientation (from the IMU) is used to generate a homography matrix that will transform ground plane points in trajectories generated by MPPI into image plane points that can be be assigned a positional cost based on the output of the deep neural network. The neural network output and associated homography matrix are used by the MPPI algorithm to plan and execute controls until another cost image is available. All of the computation for MPPI and the CNN happen on the same GPU on-board the robot.

3.3 Implementation on AutoRally

We experimentally evaluate both the top down and image plane methods with two different neural network structures. The image plane network takes in 640x480 input images and passes them through several convolution layers and 2 pooling layers, followed by a set of 6 dilated convolution layers. The top down network uses a smaller structure, as shown in Figure 4.3. The dilated convolutions allow each output pixel the full input image as its receptive field while maintaining a reasonable (128x160) output size. This significantly improves the output quality of the network. The cost-map is then taken directly as the output of the final layer without applying normalization.

Using these two network architectures, we are able to maintain low latency and a frame rate of about 10 Hz for the image plane network and 40Hz (full camera frame rate) for the top down network. Input images come directly from a PointGrey Flea3 color camera at 1280x1024 resolution. These images are downsampled to 640x512, the dataset mean is subtracted, and each pixel is divided by the dataset standard deviation. During training, the 160x128 pixel output is compared with the pre-computed ground truth cost maps obtained from GPS data. When generating ground truth cost maps, we find that an output size of

8.5m by 10.7m (15 pixels per meter) produces the best overall system performance. It was found that an L1 pixel-wise loss produced a cleaner final cost map than L2 loss. For these experiments, this loss is only computed for points within 10 pixels of the edge of the track in the ground truth image to avoid training the network to output large sections of blank space. The network was trained using the Adam [96] optimization algorithm in Tensorflow [97]. A mini-batch size of 10 images was used during training, and a small random perturbation to the white balance of each image (multiplying each channel by a normally distributed random variable between 0.9 and 1.1) was also applied. For all networks, best driving performance was achieved with training stopped at or near 100,000 iterations. This coincided with the point where testing loss on a held out dataset plateaued.

3.4 Driving Performance

Our goal in learning to regress cost maps from images is to plan and execute high speed driving maneuvers. Forward inference through the network is handled asynchronously, and the cost maps are fed to the MPPI control algorithm at approximately 10Hz for the image plane network and 40Hz for the top down network. The MPPI controller runs at 40Hz. Velocity and acceleration information is obtained from the on-board GPS-IMU system, but absolute position is not used. We use GPS derived velocity for experimental simplicity, however this velocity could be derived from visual odometry in a purely vision based system. For the top down case, the camera orientation (from the IMU) is used to generate a homography transform. The neural network output and associated homography matrix are used by the MPPI algorithm to plan and execute controls until another cost image is available.

We autonomously drive AutoRally at the Marietta track at increasingly aggressive speeds. Each method uses the same controller and vehicle physics model, cameras, and track. The form of the controller's cost remains the same, although some parameters such as exploration variance and relative cost weights are tuned slightly to optimize perfor-

Table 3.1: Testing statistics for image plane (ip) and top down (td) networks

Method	Counterclockwise travel		Clockwise travel	
	Avg. Lap (s))	Top Speed (m/s)	Avg. Lap (s)	Top Speed (m/s)
(TD) 5 m/s	16.98	4.37	18.09	4.99
(TD) 6 m/s	12.19	6.38	failure	failure
(TD) 7 m/s	10.84	6.91	11.27	6.51
(TD) 8 m/s	10.13	7.47	failure	failure
(IP) 6 m/s	14.48	5.67	failure	failure
[90]	9.74	8.44	N/A	N/A
[64]	N/A	N/A	10.04	7.98

mance. To find the limits of each method, we slowly increased the target speed from 5m/s. If the vehicle was able to performing 10 laps without intervention, the condition was considered a success as summarized in Tab. 3.1 If intervention was required, the condition was considered to have failed. Note that the friction limits of the vehicle in these turns is around 5.5 m/s, so the control algorithm has to intelligently moderate both the steering and throttle in order to navigate successfully. While this single track is a limited environment, there is still significant clutter (such as changing lighting conditions and moving distractors), making this a challenging vision problem. In addition, due to the speeds the vehicle is traveling, small network errors can lead quickly to overall system failure. As with many machine learning systems, our system is sensitive to the training data. Our dataset contains more counterclockwise examples than clockwise examples, possibly explaining the higher failure rate while traveling clockwise.

A comparison between the image plane and top down targets is performed on the Marietta track. Using the top down network produced significantly more robust, consistent, and overall faster runs than the image plane network. Using the image plane network, it was only occasionally possible to produce runs of 10 consecutive laps (at the slowest speed). Most of the runs lasted between 1 and 5 laps before intervention was required. Usually, this was due to the network not identifying a turn, which would result in the vehicle driving to the end of the track and stopping. Figure 3.2 demonstrates the difference between the two approaches as the vehicle approaches a turn. As the distance from the vehicle increases,

Table 3.2: Performance comparison for different MPPI target velocities for the top-down network, and best competing metrics using GPS position. Notice the clockwise crash due to

Metric	GPS	Image Plane	Top Down
Network Error	N/A	0.82	0.92
Average Lap Time	11.26s	14.48s	13.16s
Maximum Speed	7.75 m/s	5.67 m/s	5.62 m/s

the top-down network produced much cleaner cost maps. It is apparent that distortion from the projection significantly hinders the performance of the reprojection network.

From these experiments, it is apparent that the top down method is far superior in driving performance to the image plane with reprojection method.

We performed an ablation study in order to identify the features of the input images that play the strongest role in the generation of the cost map. We first obtain as a baseline the cost map which is generated by the network from the full input image. We then ”zero out” a block of pixels at a certain location and size by replacing all pixels within the window with the mean pixel value from the entire dataset. After mean subtraction, this block will have the value zero and will therefore not contribute to the network activations. We systematically examine the influence of different parts of the image on the prediction performance by scanning the window over the entire input image, thereby generating a set of ablation images with zeroed out blocks at different locations. For each ablated image, we compute an accuracy measure (average L1 distance for track pixels). We then construct a sensitivity map by creating an image from these accuracy measures, which each accuracy value is stored at the corresponding location where a block was ablated. Figure 3.7 shows sensitivity maps for representative input images. All sensitivity maps are normalized with zero error as black and largest recorded error as white.

This study demonstrates that the network has learned to use intuitively reasonable input features in real world experiments, and can generalize to unknown scenes in the simulation case. The network can tolerate the removal of small track regions due to ablation and still

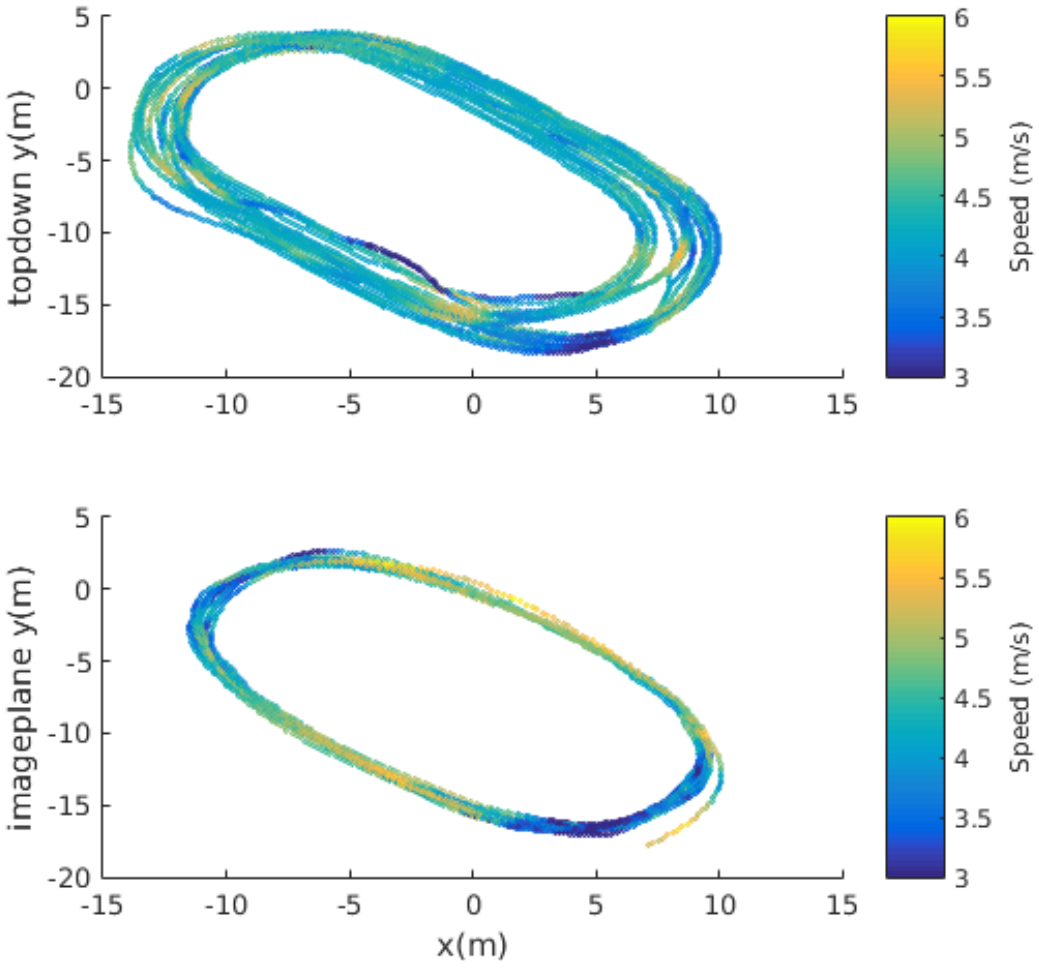
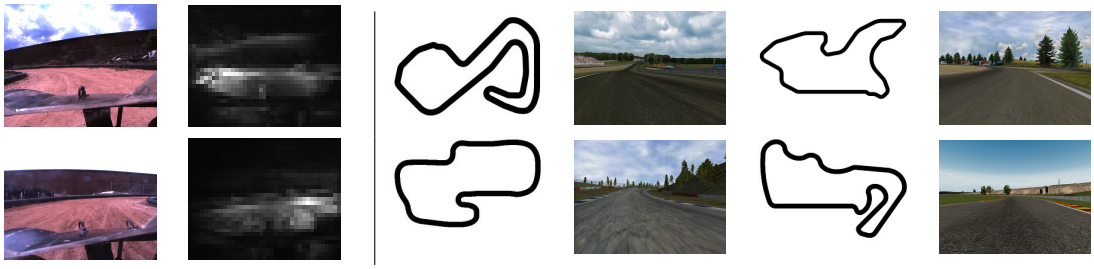


Figure 3.6: Graphs of velocity versus position. This shows the image plane cost map supports higher straight line speeds (long visible distance), but corner speeds are slower due to limited camera field of view. Top: clockwise and counterclockwise runs of neural network regressing top down cost map (overlap is due to poor ground truth pose). Bottom: Image plane cost map.



(a) Ablation study (b) Track images and screenshots

Figure 3.7: Ablation study and simulation results. (a) In the ablation study, a square window of the input image is replaced with the corresponding square of the dataset mean image. Examples shown of two input image with ablation heatmaps. It is clear in both cases that the inside of the corner is the most important feature for determining track shape, and most image clutter is ignored. (b) TORCS simulation tracks, with example training track in upper left and unseen test tracks. Track overview maps show a great deal of corner variety, and screenshots show texture variety.

produce usable cost maps. Because ablating the furthest visible point on a corner tends to most affect the output cost map, we believe this portion of the image contains the most information. These importance patterns seem to line up with the findings of [55] and [49], and points us toward future work of learning a foveation strategy from data.

In order to test the capability and generalization of this network, we train the network structure from Figure 4.3 on a dataset of images and corresponding cost maps collected from the TORCS open source driving simulator. We collect approximately 250,000 images taken from several hours of autonomous and manual driving on 4 representative tracks as training. We then validate this network by calculating validation error and tasking the network with completing several laps of three other, unseen tracks. These tracks include similar features, but have new layouts as seen in Figure 3.7(b). The validation error on the first track is 0.962, significantly better than the trivial output of entirely non-track (white) pixels of 0.896. The system is able to complete the top right and bottom left track of Figure 3.7(b). The bottom right track produced excellent error (0.961) but required intervention to drive the full track due to errors in one or two corners.

This simulation study shows that the network can generalize to unknown scenes in the simulation case, but there is still more work to perform. While these studies show that the

network can generalize, performance is still lacking even in the simulation case. By using new models and training techniques, we aim to improve this generalization.

3.5 Particle Filter

At the highest level, planning often takes place on a schematic map. We define a schematic map as a map containing only drivable surface information, such as a street map or in our case a race track layout. This is a very different representation than the sparse descriptor maps or 3-D reconstructions used for SLAM based localization. In this work, we utilize a metric map generated by surveying the centerline of our race track. A distance transform of this centerline results in the schematic map shown at the top of Fig. 3.9.

Instead of using an existing whole image or key point based SLAM system, the monocular camera images are used as the input to a convolutional neural network in order to directly regress the free space in front of the vehicle. Key point based localization on the AutoRally system is difficult due to the low vantage point, short stereo camera baseline, and many self-similar textures in the environment. Our track surface is dirt and the track barriers are black, resulting in very few useful features within range of our stereo cameras suitable for key point SLAM. Additionally, the on board cameras are subject to high amplitude vibrations during fast, aggressive driving, this breaks many SLAM algorithms that assume a low acceleration motion model.

We use a particle filter to combine the proprioceptive IMU and wheel speed sensors with the deep neural network cost map sensor. This allows us to fully utilize all available information to drive aggressively without the aid of external localization. In addition, we create a new neural network architecture that can be trained recurrently and utilizes temporal information to resolve difficult difficult or ambiguous visual situations. This improves the accuracy of the cost map prediction network over the single frame case, and greatly improves performance on difficult and ambiguous cases.

The full aggressive driving system (see Fig. 3.8) has three main components. First, a

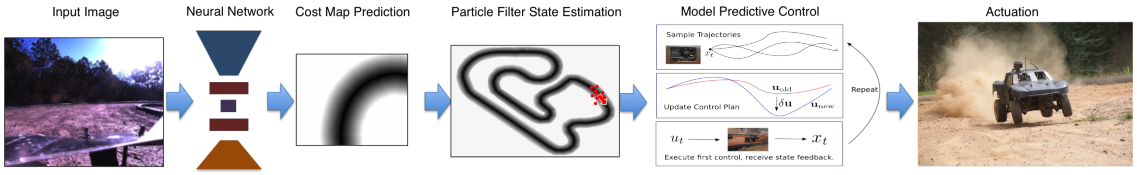


Figure 3.8: Full system diagram.

CNN is used to predict local cost maps from monocular input images. Second is the particle filter. It takes as input IMU measurements (linear accelerations and angular velocities), vehicle wheel speed measurements, and local cost map estimates from the cost map neural network. The IMU measurements are used to propagate the state forward, and particles are weighted using wheel speed and difference between cost map predictions and our surveyed cost map. Third is the AutoRally vehicle. This vehicle has all of the onboard computation and sensing needed to implement this approach and is capable of high speed, aggressive driving.

3.5.1 Implementation

Particle filters are a class of recursive Bayesian filters which attempt to approximate the state distribution with a set of samples (particles). They have the advantage of being able to handle non-linear dynamics and non-linear measurement models. However, in order to get good performance a large number of particles (numbering in the thousands) is often required.

There are many variants of the particle filtering algorithm, in this work we use the sequential importance re-sampling (SIR) particle filter. The state-space for the particle filter consists of the following 5 variables: position in the map coordinate frame (p_x, p_y), heading (ψ), and body frame forward and lateral velocity (v_x, v_y). The model predictive controller additionally uses roll information and heading derivative information, however these are passed directly from the IMU to the controller without an intermediate filtering step. The basic building blocks to the particle filtering implementation are a measurement model and a motion model:

Measurement Model

In this work, the two sensors that the AutoRally vehicle has in order to navigate are wheel speed sensors and a video stream from a monocular camera. The wheel speed sensors output a velocity estimate based on the rotation rate and diameter of the wheel, let W denote the averaged velocity estimate of the two front wheels[1], then the wheel speed measurement model is:

$$p(W|\mathbf{x}_j) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma} \left(|v_x^j| - W\right)^2, \;\; \sigma = 2.5 \tag{3.6}$$

Where $\mathbf{x}_j$ denotes the state of the j_{th} particle and v_x^j is the forward velocity. Note that the wheel speed measurements are always positive, hence the absolute value around the forward velocity.

The other sensor that the AutoRally has is a monocular camera, which provides input images to a convolutional neural network, which then outputs a prediction of the local track. This prediction can be used by the particle filter to localize the vehicle in a global schematic map via the following two-step procedure:

i) Each particle uses its current position and heading to select the local slice of the global map corresponding to the area the neural network would be making a prediction for if the vehicle were actually in that location.

ii) The local slice of the global map is compared to the actual output of the neural network, and the output of the comparison is fed into a probability distribution. The output of this distribution acts as the measurement model.

This procedure is visualized in Fig. 3.9. In this work, we use mean absolute error in order to compare the images, and feed the error into an exponential distribution. The measurement

[1]We do not use the back wheels since they slip significantly when accelerating.

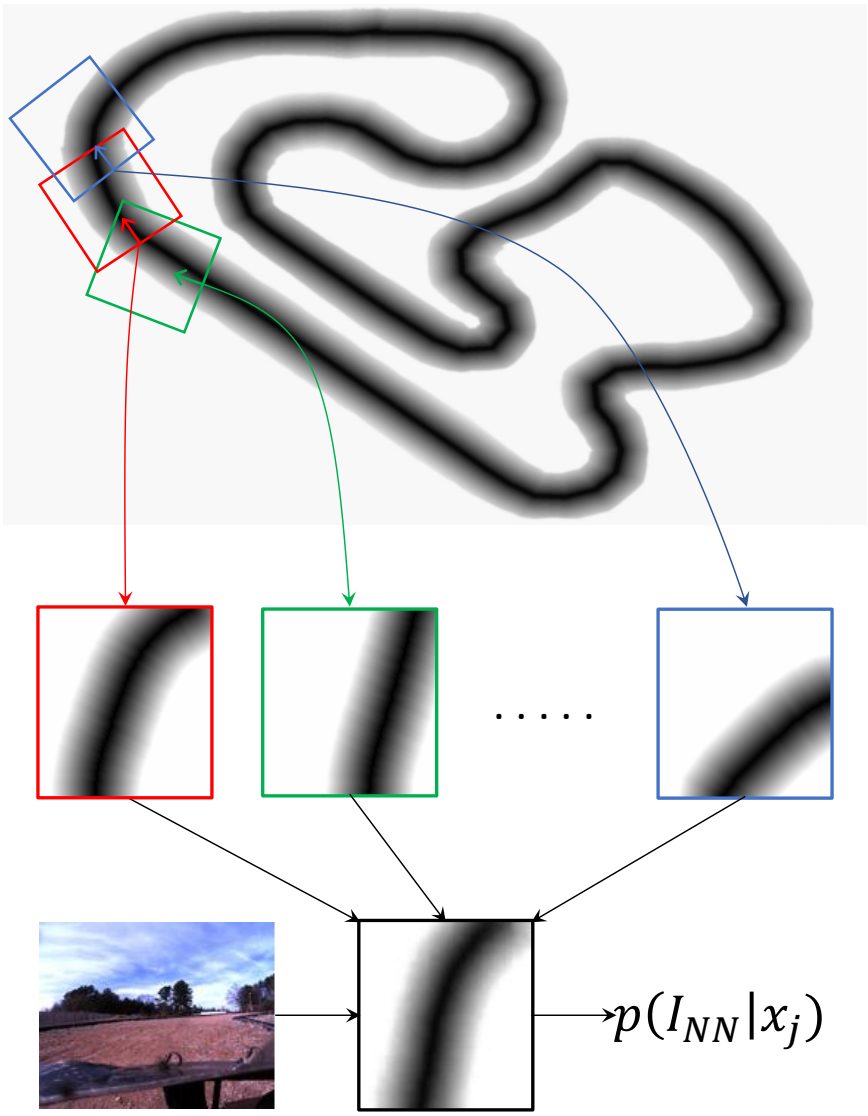


Figure 3.9: Each particle maintains a position and heading with respect to the map coordinate frame (top), this is used to extract a top-down image of the expected local track geometry (middle), and this is compared to the output of the neural network's prediction in order to compute a measurement probability.

model for the neural network predictions is then:

$$p(I_{NN}|\mathbf{x}_j) = \lambda \exp\left(-\frac{\lambda}{N}\sum_{(u,v)}\left|M^j_{\text{local}}(u,v) - I_{NN}(u,v)\right|\right) \tag{3.7}$$

where $\mathbf{x}_j$ denotes the state of the j_{th} particle, M^j_{local} is the associated local slice of the global map, and I_{NN} is the output from the neural network. Note that I_{NN} is a function of the current image and the LSTM recurrent state, which in turn is a function of all past images. The combined measurement model is then:

$$p(o_t|\mathbf{x}^j_t) = p(I_{NN}|\mathbf{x}^j_t)p(W|\mathbf{x}^j_t) \tag{3.8}$$

Motion Model

The motion model for the AutoRally is equipped with an Inertial-Measurement-Unit (IMU) which outputs accelerations (a_x, a_y, a_z) and angular velocities $(\alpha_x, \alpha_y, \alpha_z)$. These are combined with standard equations for the motion of a rigid body in order to create the following noisy motion model:

$$\begin{aligned}
\mathrm{d}p_x &= (v_x\cos(\psi) - v_y\sin(\psi))\,\mathrm{d}t \\
\mathrm{d}p_y &= (v_x\sin(\psi) + v_y\cos(\psi))\,\mathrm{d}t \\
\mathrm{d}\psi &= \alpha_z\mathrm{d}t + \sigma_\psi\mathrm{d}w \\
\mathrm{d}v_x &= a_x\mathrm{d}t + \sigma_{v_x}\mathrm{d}w,\ \mathrm{d}v_y = a_y\mathrm{d}t + \sigma_{v_y}\mathrm{d}w
\end{aligned}$$

Where $\sigma_\psi = 0.275$ and $\sigma_{v_x} = \sigma_{v_y} = 0.75$. Note that this model is different than the motion model that the model predictive controller uses to control the vehicle. Future IMU measurements are obviously not available for the MPC predictions, so this model cannot be used for MPC. The model that the MPC controller uses could be used by the particle filter in place of this rigid body dynamics model, and could potentially lead to improved per-

formance by introducing more constraints on the dynamics. However, this model provides adequate performance and is very cheap to compute.

In our particle filter implementation, the stochastic motion model propagates forward at 200 Hz with standard EulerMaruyama integration, measurements are processed at 20 Hz, and particle re-sampling occurs at 5 Hz. All of the motion and measurement models are implemented as CUDA Kernels on the GPU, which is necessary since every measurement update requires computing the difference between two images (the local image patches are 35x25 pixels). It is possible to run both the measurement and re-sampling loop at faster rates, however doing so did not lead to improved performance and other processes (the controller and neural network) also require GPU resources. We used 6400 particles, and computed the final state estimate as the mean of the particles.

CHAPTER 4

NEURAL NETWORK ARCHITECTURES

4.1 Introduction

Figure 4.1: AutoRally vehicle navigating a bump on the track at high speed during testing of the particle filter method, demonstrating its aggressive maneuvering capabilities.

We have now demonstrated two effective ways to use cost map prediction for aggressive driving on a dirt track. Both methods use cost maps directly regressed from input image frames to feed model predictive control. However, the structure of this cost map prediction network has a large impact on overall system performance. In this section, we introduce network architectures that allow increased performance and generalization, as well as describing datasets collected for this training.

We compare three primary types of neural networks, flat, encoder-decoder, and recurrent. The flat network, used in the comparisons in the previous chapter, is fully con-

volutional. It learns a series of convolutional transformations of the input, and the final convolution directly regresses the cost map without any fully connected layers. This architecture gains most of its representational power from dilated convolutions, allowing very large receptive fields at each output pixel. The encoder-decoder architecture improves generalization by using convolution and pooling to compress the input to a small intermediate representation, and fully connected layers and transposed convolution [98] to regress a local cost map from this compressed representation. Finally, we add recurrence to this encoder-decoder architecture to integrate temporal information and further improve performance.

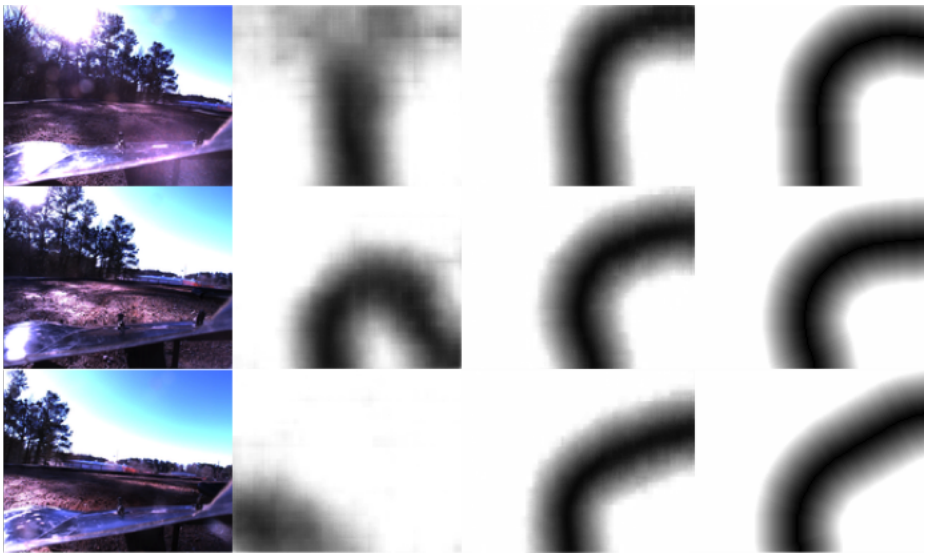


Figure 4.2: Failure case for single frame network. From left to right: input image, single frame network output, lstm output, ground truth cost map

In the case of the fully connected network and the non-recurrent encoder decoder, treating each input frame independently leads to a very challenging learning problem. This stems from the limited field of view and low vantage point of our camera, which makes it difficult to generate cost maps that extend sufficiently far in front of the vehicle. This problem is exaggerated by the high speed of travel. Our solution to the limitations of single frame cost map prediction is to incorporate recurrence in the form of an LSTM model, making it possible to exploit the temporal continuity of the track via the structure of the on-board camera video. Fig. 4.2 illustrates both the difficulty of predicting a high quality

cost map from a single image under challenging conditions, and the significant improvement that arises from incorporating recurrence into the model. The single-frame cost map predictions shown in the second column are significantly less accurate than the cost maps produced by the LSTM model (third column), which is described in Section 4.2.

4.2 CNN Architectures

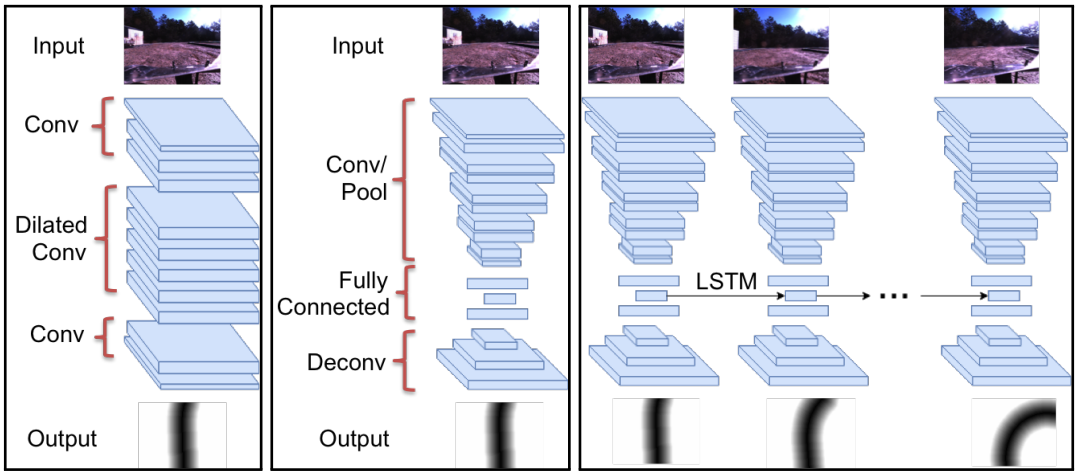


Figure 4.3: Network architectures with input and training targets. Left: Fully convolutional neural network architecture. Dilated convolutions are used to increase receptive field. Center: Encoder decoder neural network architecture. Right: Encoder decoder architecture trained recurrently, with an LSTM replacing the bottleneck fully connected layer.

Our CNN architecture is constrained to run in real time on the low power Nvidia GTX1050 GPU available on our platform, along with the model predictive controller and particle filter. Because other processes are competing for GPU resources and we require the neural network to run at full camera frame rate (40 Hz) to lower system latency, this constrains us to smaller, simpler architectures. The CNN directly produces a dense cost map in the egocentric frame, with the current vehicle position at the bottom center of the image. An overview of the comparison architectures is shown in Fig. 4.3.

The encoder-decoder architecture improves over the flat network in several ways. First, since the input image is reduced to a small hidden state, we can easily add recurrence to the

network to take advantage of temporal information. This allows resolution of some cases which are ambiguous or difficult in the single frame case. We compare the encoder-decoder and recurrently trained encoder-decoder cases to highlight the performance improvement of using video data. Second, because the network output is the result of transposed convolution of a small hidden state, the resulting cost map is much cleaner than the output of fully convolutional network, and unseen input images tend to produce more reasonable output, even if incorrect. In contrast, the flat network tends to produce only noise for input images far from the dataset distribution, limiting its ability to generalize. This behavior is shown in Fig. 4.4. We further explore the generalization power of the bottleneck architecture with leave one direction out training experiments at the CCRF track. These results are presented in Section 4.5.2.

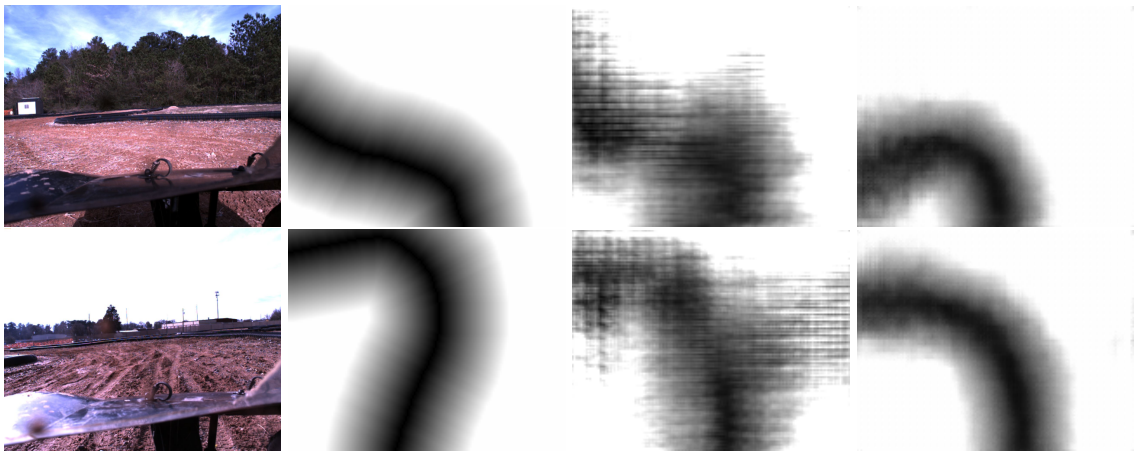


Figure 4.4: Bottleneck architecture generalization. Two cases where the bottleneck architecture produces slightly incorrect, but crisp, results. Left to Right: Input image, Ground truth image, Fully convolutional output, Bottleneck output

Third, the fully convolutional network constrains the output cost map size to match the input image size or an integer multiple of it. However, the encoder-decoder architecture can have arbitrarily shaped output cost maps. We use this ability to further optimize the output of the network to the particle filter. Previously, the output of the network was set to an area of 10.6m wide by 8.5m high (160x128 at 15 pixels per meter). However, this size is optimized for direct planning and control on the output cost map. The particle filter

uses a 5m wide by 7m high crop of this output to achieve optimal performance. By using the ability of the encoder-decoder to output arbitrarily shaped cost maps, we can directly output a 5m x 7m image (40x56 pixels) at 8 pixels per meter. This size was empirically determined to improve performance for the particle filter. We find that this significantly increases the performance of the particle filter as shown in Tables 4.4 and 4.5.

Input images come directly from a PointGrey Flea3 color camera operating at 1280x1024 resolution. These images are downsampled to 160x128 and the images are normalized by subtracting the dataset mean and dividing by the dataset standard deviation. During training, the cost map output is trained to minimize the L1 distance to the pre-computed ground truth cost maps obtained from GPS data

$$MAE(\theta, U_t) = \sum_{(u,v)} |I_{u,v} - \hat{I}_{u,v}(\theta, U_t, S_{t-1})| \tag{4.1}$$

where the L1 error MAE at input image U_t is a function of the CNN parameterized by θ. This CNN is also a function of the previous hidden state S_{t-1}. The error is the difference between the estimated cost map $\hat{I}$ and true cost map I summed over all pixels u, v in the image.

L1 loss is utilized over L2 loss due to its outlier rejection. Because it is minimizing posterior expectation of the median instead of the mean, it is less sensitive to outliers in our dataset. However, the L1 loss is less stable during training and tends to fall into a local minima where the dataset median is predicted regardless of the input, making hyper-parameter tuning more difficult. During training, recurrent networks often required weight initialization from single-frame trained networks in order to converge to non-trivial minima.

All networks are trained using the Adam [96] optimization algorithm in Tensorflow [97]. A mini-batch size of 16 images was used during training, and a small random perturbation to the white balance of each image (multiplying each channel by a normally distributed random variable between 0.9 and 1.1) was also applied. For all networks, training

was stopped after the validation error plateaued.

Our image datasets are statically split into validation and training sets, with some validation data being taken from days which are not included in the training set. Additionally, we hold out 8 full 3-laps runs at the CCRF testing site, to act as a test set for both particle filter accuracy and pixel wise neural network accuracy. This data is also from a day not included in our training set. Validation set accuracy is used to determine when to stop training. Networks we train generally are low capacity, on the order of 100's of thousands of parameters, in order to maintain real time performance. This tends to prevent overfitting, so training is stopped when validation set accuracy plateaus. The networks are trained using the Adam [96] optimization algorithm in Tensorflow [97]. A mini-batch size of 16 images was used during training.

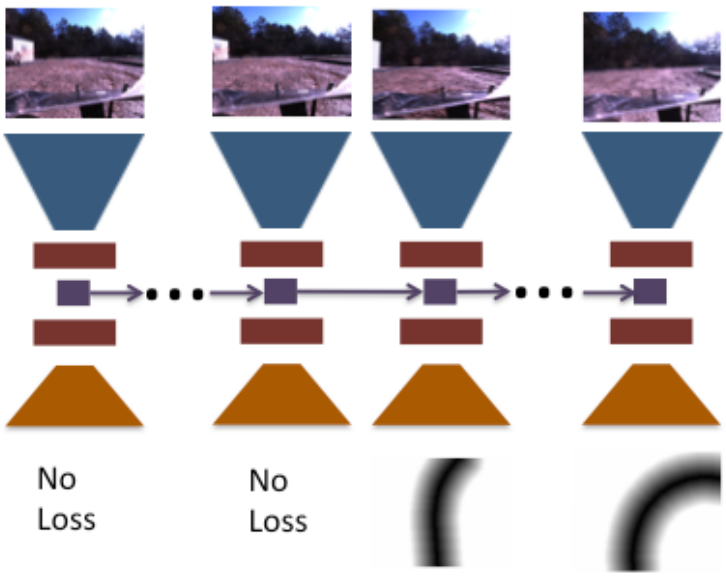


Figure 4.5: During training, recurrent neural networks only begin accumulating loss after seeing part of the sequence of input images.

In order to train the recurrent networks, the hidden state needs to be initialized. We initialize the hidden state to zero and allow the network to "burn in" for some number of frames without penalizing its output. This scheme is illustrated in Fig. 4.5. We find that

allowing the network to run for 8 frames without penalizing its output, and then accumulate training error for the next 8 frames produced the best results, for a total of 16 input images seen per training sample. We found that training over longer sequences did not significantly improve testing accuracy or final particle filter performance, and made training more time consuming and more prone to oscillation due to exploding gradients or getting stuck in local minima. We find relatively low sensitivity to the length of time given for burn-in.

Architecture Details

Add all of the architecture details here

4.3 Neural Network Dataset Results

4.3.1 CCRF

All experimental results for these comparisons are collected using our 1:5 scale AutoRally vehicle at the CCRF track, shown in Fig. 4.6. This challenging track includes turns of varying radius including a 180 degree hairpin and S curve, and a long straight section. All results are the vehicle driving autonomously using the camera for localization, with no global position of any type from the GPS. The monocular color camera, IMU, and wheel speed sensors are the only sensors used except for the direct CNN map usage case, where vehicle velocities are derived from GPS in order to simplify the software.

We compare our recurrent encoder decoder architecture to two different single frame networks using the training objective (L1 pixel error), dataset particle filter recovered position error, and on-policy driving performance using the MPPI controller. We compare the performance for both the fully trained and leave-one-direction-out case.

Because the recurrent architecture was trained over sequences of data, the network was able to learn several interesting temporal patterns in the input data. It learned to smooth the cost map prediction from frame to frame, so output does not jitter as much as the single frame network. It also learns to integrate information over multiple frames in areas where it

Figure 4.6: Test track for physical vehicle experiments. This track includes a variety of turns, and is very challenging for the visual navigation system.

is difficult to see where the track goes, such as Fig. 4.2. This allows the LSTM to produce much better results in difficult or ambiguous areas that cannot be easily interpreted in a single image.

Training data was gathered during the course of normal vehicle testing at the Georgia Tech Autonomous Racing Facility over the course of approximately one year, shown in Fig. 4.6. Because this system has the ability to learn from any data, on or off policy, we utilize data collected from various experiments. In total, we collect approximately 90k samples. These samples are broken into about 75k training samples and 15k testing samples. The testing samples are taken as full held-out contiguous runs to allow recurrent network testing. In addition to the testing samples, we report test error on a corpus of data that includes 8 test runs performed using the particle filter with 3 laps each.

Table 4.1: Average L1 pixel accuracy

	Flat	ED	ED-R	ED-S	ED-R-S
Train	94.66	93.66	96.48	92.69	96.09
Val	92.29	91.64	92.54	88.13	89.64

ED: Encoder Decoder. R: Recurrent. S: 40x56 output

Average training and Validation accuracy for the networks is shown in Table 4.1. Since there is no straightforward way to normalize these results for map scale and the number

of pixels available, the accuracy is not directly comparable between the three large output (128x160) networks and the two small (40x56 output) networks. The LSTM networks achieve significantly better validation error due to learning to integrate information temporally before producing a result. It is apparent from Fig. 4.2 that this is at least partially due to the networks ability to integrate information over time and correctly identify visually challenging frames. Single frame networks are not able to identify the track in these challenging cases.

4.4 Direct Driving Results

Table 4.2: Lap times for Direct Driving Method at 6 m/s target speed

Method	Flat	ED	ED-R	ED-S	ED-R-S
Direct Driving	37.3	37.8	39.1	NA	NA

ED: Encoder Decoder. R: Recurrent. S: 40x56 output

We first provide a comparison of these methods via the direct driving algorithm, allowing MPPI to plan directly on the output of each image as described in 3. Both small networks, which output 5m x 7m regions, are unstable using this method and crashed almost immediately. This is due to the output image size being too skinny for MPPI to plan a reasonable path. The summary of lap times for the other methods are show in Table 4.3. At moderately aggressive speeds, all networks are able to reliably and repeatably complete 3 consecutive laps of the challenging CCRF track.

4.5 Particle Filter Results

Localization performance of the particle filter, as well as real-world performance of the full system, was measured by driving the vehicle at a 6 m/s target speed for 3 laps using each of 5 neural networks (approximately 100s of driving for each condition). These 5 networks are trained in 2 different ways. First, using all available data. This includes clockwise

and counterclockwise training data from 7 days of testing spread over a period of approximately a year. Second, we perform leave-one-direction-out (lodo) testing by training neural networks on only the subset of data where the vehicle is traveling clockwise. All testing and validation is performed with the vehicle traveling counterclockwise. Average position error versus the GPS derived ground truth is reported for both the on-policy case where the vehicle is driven using the pose estimate and the off policy case where the filter is run off-line on recorded data and the average position error is calculated.

Table 4.3: Lap times for Direct Driving and Particle Filter at 6 m/s target speed

Method	Flat	ED	ED-R	ED-S	ED-R-S
Direct Driving	37.3	37.8	39.1	NA	NA
Particle Filter	35.6	34.8	35.9	34.5	35.3

ED: Encoder Decoder. R: Recurrent. S: 40x56 output

In order to test the particle filter method against previous work, we estimate the full state of the vehicle, position and velocity, with the particle filter and the same networks used for the direct driving network with a target speed of 6 m/s. However, in these experiments, we allow MPPI to plan on the pre-surveyed map used by the particle filter for localization. All networks produce sufficiently accurate results to easily complete the track at a 6 m/s target speed.

4.5.1 Full Dataset Training

In the full training case, all network architectures were able to successfully drive the vehicle in the on-policy condition. These networks are trained from scratch on the full CCRF dataset. The particle filter position errors using these errors are summarized in Table 4.4. The small recurrent encoder-decoder achieved the best average position error when tested against recorded data for all 8 3-lap on policy runs. It significantly out-performs the non-recurrent encoder-decoder, demonstrating the benefits of recurrence. Figure 4.7 shows the position error as a function of ground truth track position. We can see that the error for the

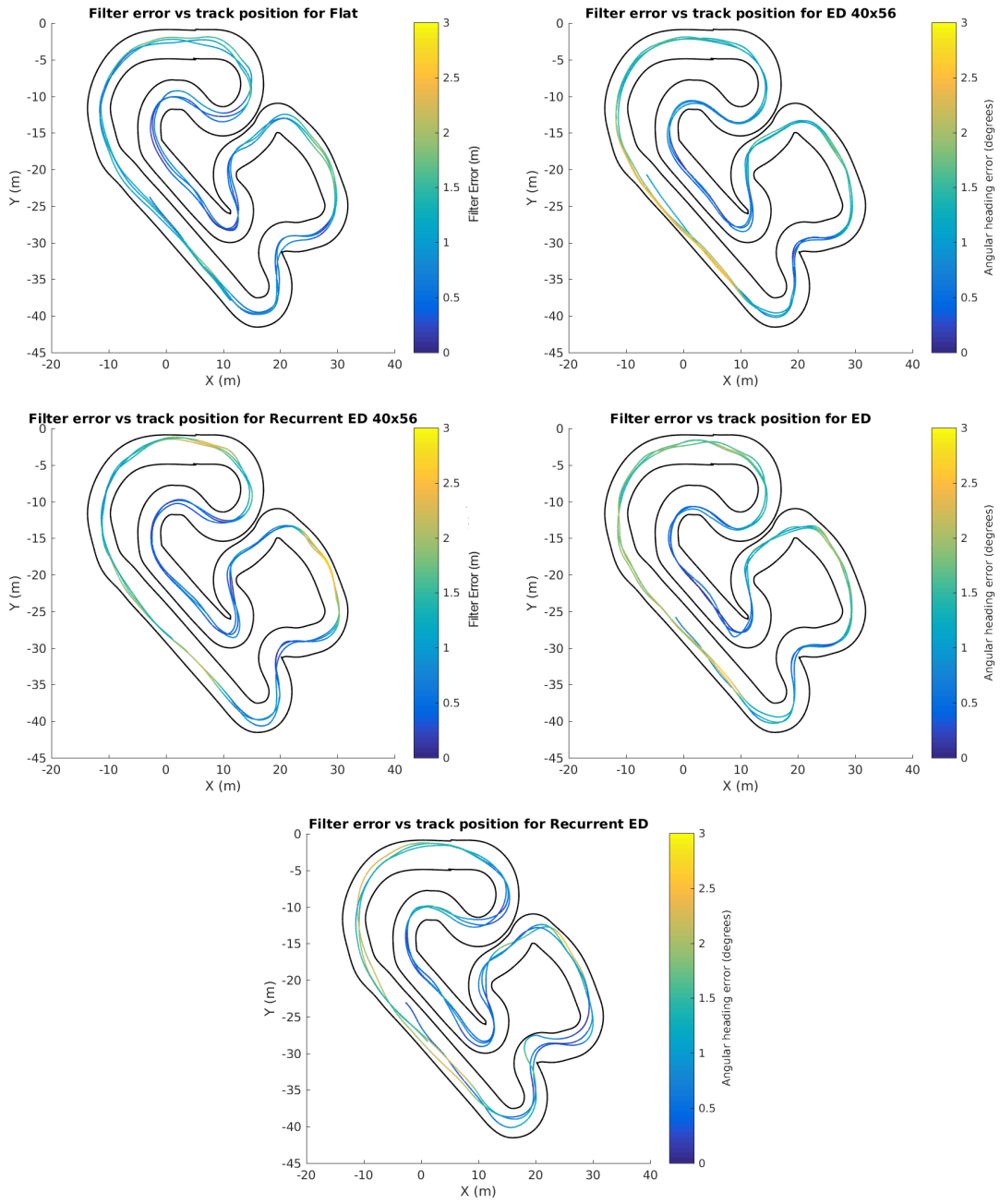


Figure 4.7: Particle filter position estimate error plotted vs ground truth position. For each network, MPPI was run with a target speed of 6 m/s using the particle filter pose estimate as its pose source. MPPI planned using position, velocity, and orientation from the particle filter and track cost from the pre-surveyed map.

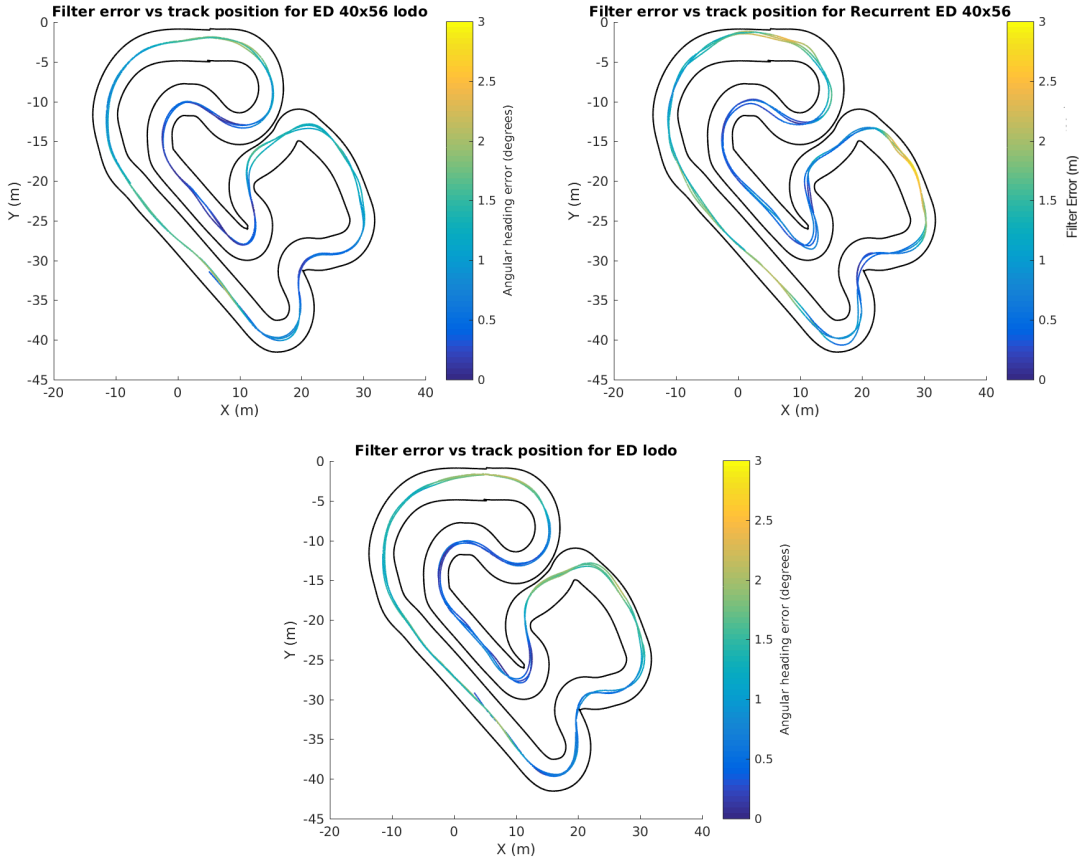


Figure 4.8: Particle filter position estimate error plotted vs ground truth position for leave one direction out experiments. For each network, MPPI was run with a target speed of 6 m/s using the particle filter pose estimate as its pose source. MPPI planned using position, velocity, and orientation from the particle filter and track cost from the pre-surveyed map.

Table 4.4: Particle Filter Position Error for Full Dataset Trained Neural Networks

Dataset↓	ED	ED-S	Flat	ED-R	ED-R-S
ED	1.21	1.12	1.01	1.19	1.0
ED-S	1.27	1.19	1.02	1.11	1.07
Flat	1.04	0.94	0.92	0.98	0.89
ED-R	1.08	0.99	1.05	1.01	1.05
ED-R-S	1.17	1.01	0.88	1.09	1.09
ED-lodo	1.22	0.91	1.45	1.06	1.05
ED-S-lodo	0.92	0.86	0.87	0.88	0.84
ED-R-S-lodo	0.93	0.96	0.81	1.06	0.80
Average	1.11	1.00	1.00	1.05	0.97

recurrent network tends to be concentrated in shorter sections of the track, and that there are some track sections that are difficult for all networks.

4.5.2 Leave One Direction Out

Table 4.5: Particle Filter Position Error using Leave One Direction Out Neural Networks

Dataset↓	ED	ED-S	Flat	ED-R	ED-R-S
ED	1.18	1.03	1.06	0.954	1.04
ED-S	1.15	1.03	NA	0.779	0.99
Flat	0.981	0.904	0.812	NA	0.854
ED-R	1.11	1.12	0.91	0.823	0.986
ED-R-S	NA	NA	NA	1.08	1.19
ED-lodo	1.05	1.02	1.23	1.46	1.04
ED-S-lodo	0.96	0.90	1.11	0.61	0.84
ED-R-S-lodo	0.93	0.91	0.97	0.62	0.85
Average	1.05	0.99	1.02	0.90	0.97

ED: Encoder Decoder. R: Recurrent. S: 40x56 output.
NA: failed to initialize on policy

Leave one direction out testing is performed to show the limits of generalization of the particle filter method. Each network is trained only on data from clockwise traversals of the track, and tested running counter-clockwise. None of these neural networks is able to successfully traverse the track using the direct driving method when not trained on the counter-clockwise direction. However, when integrating IMU and wheel speed data on a pre-surveyed metric map using the particle filter, we are able to generalize with some network architectures.

We find, in the leave-one-direction-out case, that the encoder-decoder and recurrent encoder-decoder networks are better able to generalize to this new environment and maintain a sufficiently accurate pose than the flat network, as demonstrated in Table 4.5. In the on-policy case, the flat network and the large recurrent encoder-decoder were unable to maintain an accurate pose, and caused the vehicle to crash into a barrier before completing 3 laps. Both small output networks and the large non-recurrent encoder-decoder were able

to successfully drive the vehicle at a 6m/s target speed for 3 laps. In the off-policy case, the particle filter using the flat network failed to initially converge in 2 of 5 cases. The small recurrent encoder-decoder successfully initialized the particle filter and tracked pose through all datasets, with the encoder decoder networks achieving lower overall error than the flat network.

4.5.3 Performance Limits

In order to test the limits of our overall system, we tasked MPPI with traversing the track as quickly as possible. We do this using the particle filter position estimate with no GPS input, and MPPI planning on a pre-surveyed metric map. Using this method, we set a lap time of **27.9s at 12.2 m/s (27.3 mph) top speed**. Figure 4.9 shows that the algorithm is able to reliably perform 5 laps while sustaining high speeds and high slip angles. For comparison, our previous best published lap time at this track is 32s at 8.5 m/s maximum speed, and the best lap in all training data is 29.4s at 10.4 m/s. This method is able to consistently push the limits of the vehicle and execute aggressive maneuvers using only a monocular camera, IMU, and wheel speed sensors.

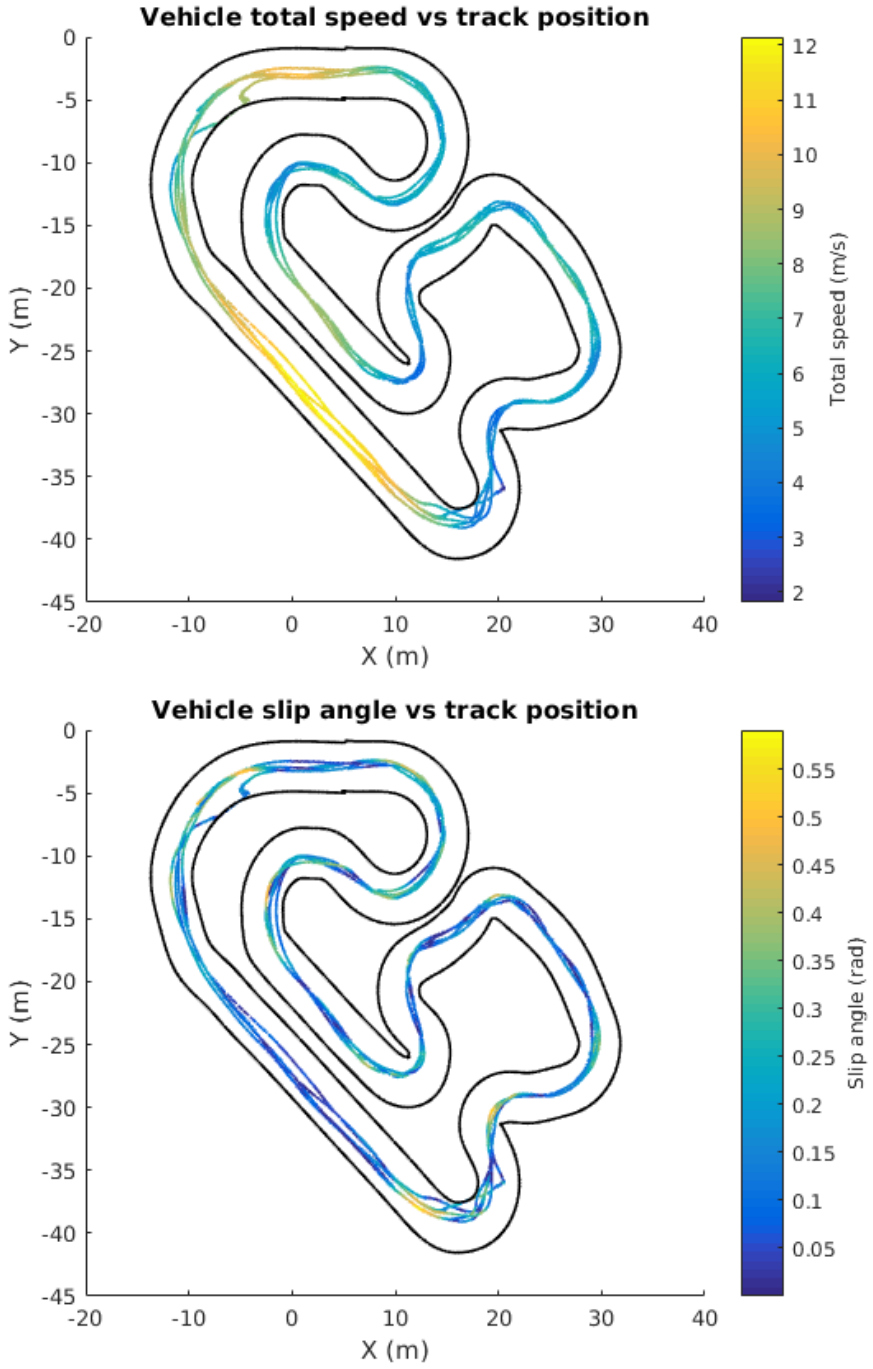


Figure 4.9: Vehicle speed and slip angle and speed as it traverses 5 laps at the limits of performance. The vehicle needs to be able to drive from 2 m/s to 12 m/s, at slip angles up to 32 degrees, in order to maximize lap times.

CHAPTER 5
ATTENTION

5.1 Introduction

In this chapter we will explore the emergent behavior of attention mechanisms in convolutional neural networks and the additional generalization power these attention mechanisms grant the system. Thus far we have shown the effectiveness of our top-down cost map representation for aggressive driving, the benefit of using temporal information in predicting from video results, and the generalization power of compressing the neural network representation before making a final prediction. In this chapter, we will show how the use of these tools and techniques in combination with end to end learned visual attention allows us to effectively use low fidelity simulation data to help generalization to unseen tracks. Additionally, we show that this end to end learned attention mechanism learns to attend features similar to those a human attends to while performing the same task.

Because we use the softmax function to encourage sparsity in the attention image, this attention acts as a regularizer. Typically, regularization of neural networks is accomplished with dropout or weight decay. Neither of these schemes is able to take advantage of the spatial or temporal structure of the problem. In the case of dropout, the regularization is purely stochastic. While this encourages redundancy, it does not use the structure of the problem. Weight decay applies regularization in the space of neural network parameters, but again does not encode the problem structure in this regularization.

Attention in humans is a well studied field, with a great many models available and many human studies using gaze tracking to compare with these models and tune parameters. Human attention while driving is also well studies, with the advent of high fidelity simulators [55] and compact, portable eye tracking [49] allowing in-situ studies of human

gaze while performing the task of driving. However, human attention while driving aggressively, and computational models of this attention, are less well understood. Some work has been performed studying human attention of race car drivers [50], however, there are few computational models of this attention.

Instead of using a model to try and mimic or explain human attention, we learn an attention model end to end for the task of predicting top down cost maps. In our earlier work, we have shown that this representation is well suited to aggressive driving on a race track, so it is a natural intermediate representation for learning attention for aggressive driving. Because this model is not constrained or trained in any way to mimic human attention, there is no reason for it to attend to similar regions of the image. However, we find that the generated soft attention map uses features similar to the features humans use. This supports the theory that humans attention strategy is an emergent property of the data and task being performed, not simply a byproduct of the human foveal attention mechanism. We also show that this attention mechanism helps generalization performance to novel environments and learning using simulation data. This provides evidence for the usefulness, beyond human eye structure, of attention.

5.2 Structure

We introduce a new neural network structure in order to learn an attention mechanism from video. This structure builds from our previous work using recurrence and representation compression in order to retain these benefits. We retain the encoder-decoder neural network structure which we have shown works well for generalization and produces crisp output cost maps. We retain the single-layer recurrence that was also shown to work well at learning video patterns. We depart from this structure by changing from a standard LSTM at the fully-connected bottleneck to a Convolutional LSTM (Conv-LSTM), as introduced in [99], in feature maps of the encoder portion of the network, with a resolution of 40x32. Additionally, we add a soft attention mechanism, as introduced in [11], before the input to

the Conv-LSTM.

The Conv-LSTM is a recurrent neural network with the same input gate, output gate, forget gate, and hidden state structure as a standard LSTM. However, instead of using fully connected layers which have a difficult time exploiting spatial information, the input to hidden and hidden to hidden transitions convolutions. This allows the hidden layer to maintain its spatial structure and all transitions can maintain the advantages of convolutions. The choice of a convolutional LSTM is motivated by the temporal consistency of the input frames, and the desire for temporal consistency in the attention produced by the network. Because we have high frame rate video as the input to our learning system, it is reasonable to assume each frame is closely related to the previous frame spatially. Given this insight, a Conv-LSTM is the logical choice for propagating spatial information and producing spatially and temporally consistent attention.

We implement an attention structure using a softmax transformation over the spatial features presented at the n^{th} convolution layer. This attention image is produced with the softmax function

$$A^{u,v} = \frac{e^{x^{u,v}}}{\sum_{u,v} e^{x^{u,v}}} \tag{5.1}$$

where A is the attention weighting at each feature map location u, v in $\mathbb{R}^{U,V}$, where U, V are the image width and height and x is a prediction vector. This prediction vector at time t, x_t, is a projection of the hidden state of the convolutional neural network h at time $t-1$:

$$x_t = W_{attn} * h_{t-1} \tag{5.2}$$

where h lives in $\mathbb{R}^{s,U,V}$ where s is the hidden size. W is the attention projection kernel which is convolved with the hidden state h. The attention image is then broadcast to the number of feature maps produced by the n^{th} convolutional layer and element-wise multiplied to produce attention weighted inputs to the convolutional LSTM.

This structure gives us an overall neural network structure as shown in Fig. 5.1 and

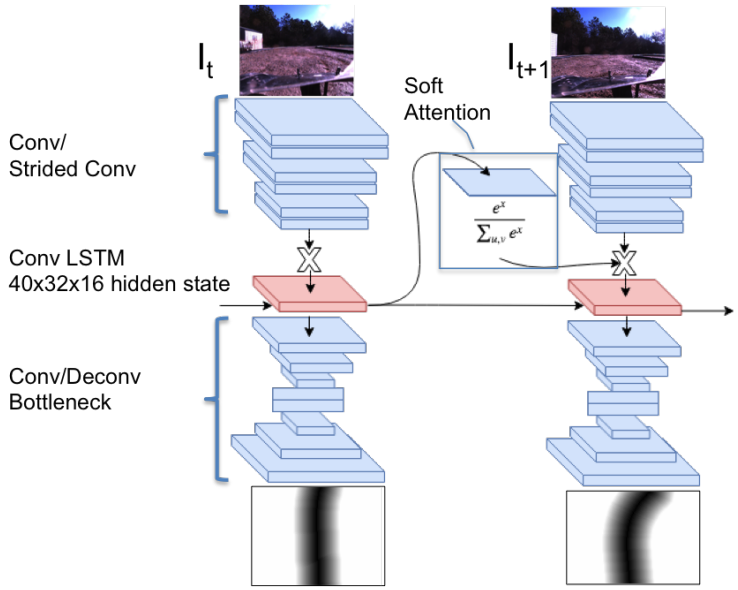


Figure 5.1: Neural network structure with convolutional LSTM recurrence and attention

detailed in Tab. 5.1. This architecture is considered "soft" attention, as opposed to "hard" attention, because there is no requirement for the network to chose only a single point or area to process. Instead, the softmax operation enforces that all weights sum to one, meaning we can interpret this attention image as a probability that an image region (defined by the receptive field at the n^{th} layer) will be useful for predicting the final driving map. In practice, the softmax operation causes sparsity in the attention image, forcing the network to only look at useful regions of the image.

5.3 Human Attention Comparison

There is a large body of work addressing the problem of modeling human attention in either bottom up or top down fashion. Some of these works provide end to end learned, computational models of human gaze. However, these are generally not embodied models and are learned to provide some type of attention on an image based task. Other embodied works generally provide models of human gaze, but these models are optimized to predict human gaze. In contrast, we provide an embodied solution with a real robot operating in

Table 5.1: Neural Network Structure for Attention Neural Network

	Filters	Filter Shape	Stride	Size
Conv1	32	7x7	1	128x160
Conv2	64	3x3	2	64x80
Dropout				64x80
Conv3	64	3x3	1	64x80
Conv4	64	3x3	2	32x40
Dropout				32x40
Conv5	64	3x3	1	32x40
Conv6	64	3x3	1	32x40
Conv7	1	1x1	1	32x40
Attn Map		3x3		32x40
Conv LSTM		3x3		32x40x16
Conv8	16	3x3	2	16x20
Conv9	16	3x3	2	8x10
Conv10	16	3x3	1	8x10
FC				64
FC				160
ConvTranspose1	16	3x3	2	8x10
ConvTranspose2	16	9x9	4	32x40
ConvTranspose3	16	9x9	4	128x160

the world with and end-to-end learned gaze model designed to optimize task performance, not predict human gaze.

In order to compare this model with human gaze, we must collect gaze samples of a human performing the same task. In order for the comparison to be as fair as possible, we allow the human to view images from same onboard camera used by the neural network to predict track in front of the vehicle. These images are streamed in real time back to a monitor, where they can be displayed for a human operator. This monitor is equipped with a real time gaze tracker to record the gaze location of the human operator. The human is given the task of driving quickly around the track while not crashing into the barriers. In order to minimize the learning curve for this experiment, we record gaze for three researchers who vary in experience but are all familiar with this vehicle and its dynamics.

Because the human gaze is recorded as a series of points on a video stream, we need to convert this data to a heat map in order to compare to the probability map generated from

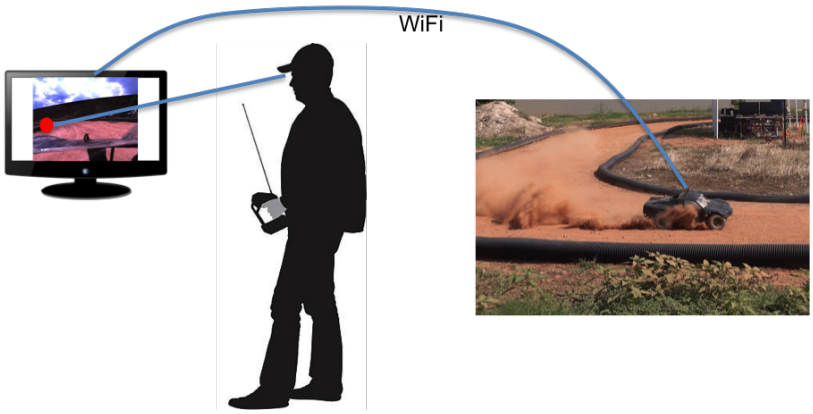


Figure 5.2: Experimental setup to measure human gaze. Images from the camera onboard the vehicle are transmitted via WiFi to a monitor where the human drives and their gaze point on the monitor is recorded.

the neural network. To do this, we take advantage of the similarity of images from nearby locations on the track. Using the ground truth poses recorded on the vehicle, we select only image frames from a limited region of the track, as shown by the blue dots in the left hand side of Fig. 5.3 Since these frames are from a visually similar region, with the human performing a similar task, we can combine the images and gaze locations to form a heat map that can be compared with the heat map produce from the neural network.

Because the gaze patterns of the neural network are not limited to a single region, a direct overlap metric is not very informative. However, as shown in Fig. 5.3, we can qualitatively see that the neural network learns both an intuitively reason gaze pattern (watching the track barriers) and a gaze pattern that lines up well with a human. As predicted by prior work such as [50] and [100], the human drivers generally watch the apex of the turn while approaching and executing a corner, and a point in the middle of the track when exiting a corner and driving down a straight. The neural network gaze often has a strong local maxima at the apex of a corner when cornering, and looks at both the left and right barriers when driving down a straight. This behavior is reasonable given the suggestions by previous research [100] that drivers may use distance from lane markings as a cue when fixating in the center of the road.

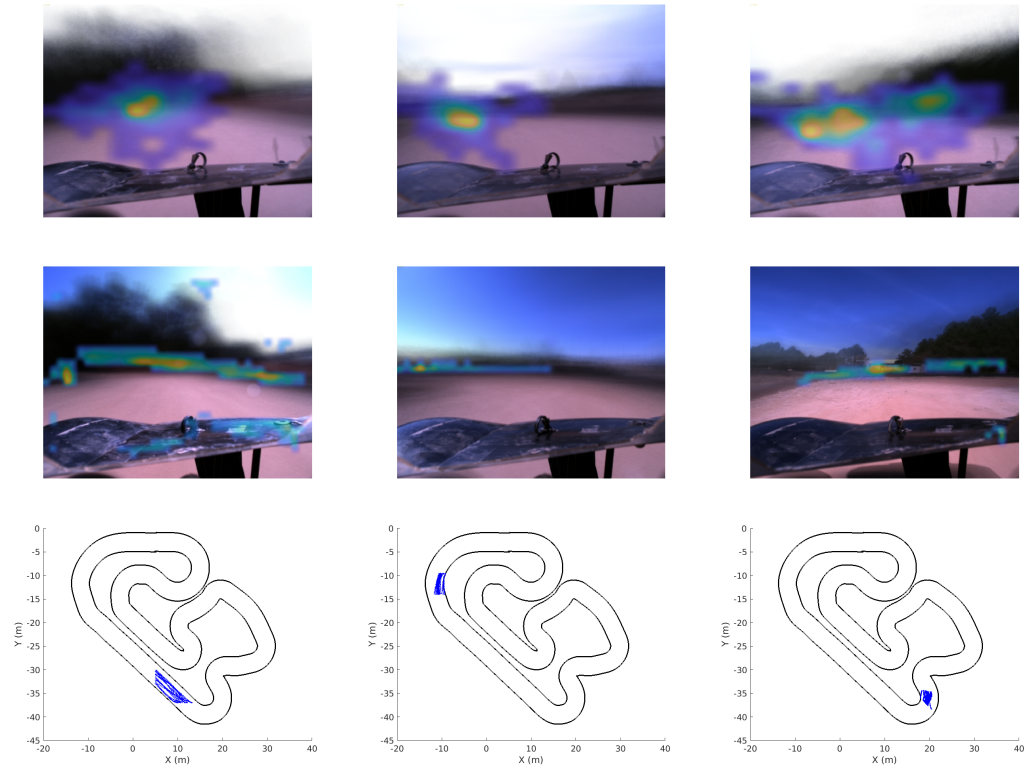


Figure 5.3: Human and CNN attention location comparison. Top row: Human gaze heatmaps. Gaze locations from all participants are averaged over a small track location leading up to a corner. Middle row: CNN attention heat map from same track location. Bottom row: Locations for gaze averaging plotted on track map.

5.4 Simulation training

Because there are many distractors in our environments, and background objects such as buildings that are stationary across all data sets, generalization to new environments is difficult. Stationary background objects are often used by the neural network because they remain in the same relative position across all data. Because we wish to generalize between the Marietta and CCRF tracks, the width of the track becomes an issue. The Marietta track is approximately 4 meters across, while the CCRF track is 5 meters across. This significantly changes the visual queues needed to find the edges of the track. With the flat and bottleneck architectures, generalizing from the simpler Marietta track to the more complex CCRF track was not possible.

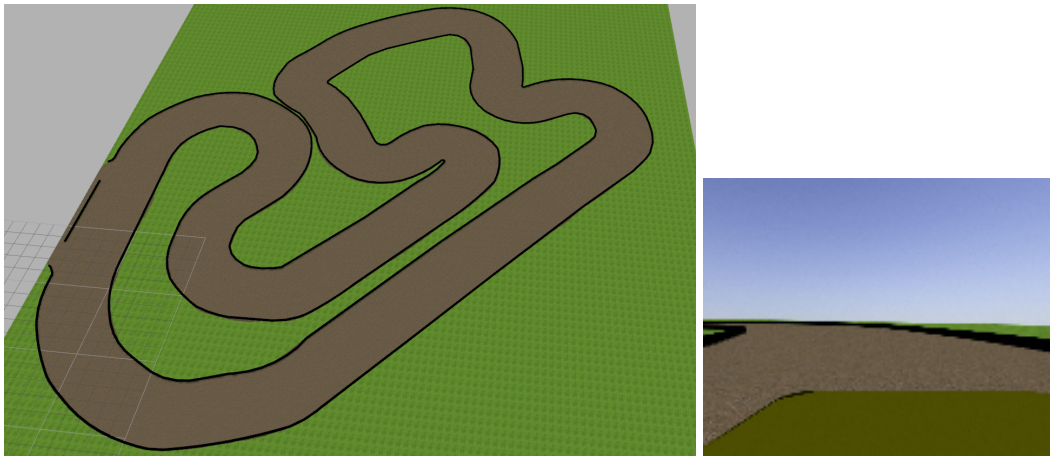


Figure 5.4: CCRF simulation world. Left is an overview of the track and right is an example image from the vehicle perspective used for training.

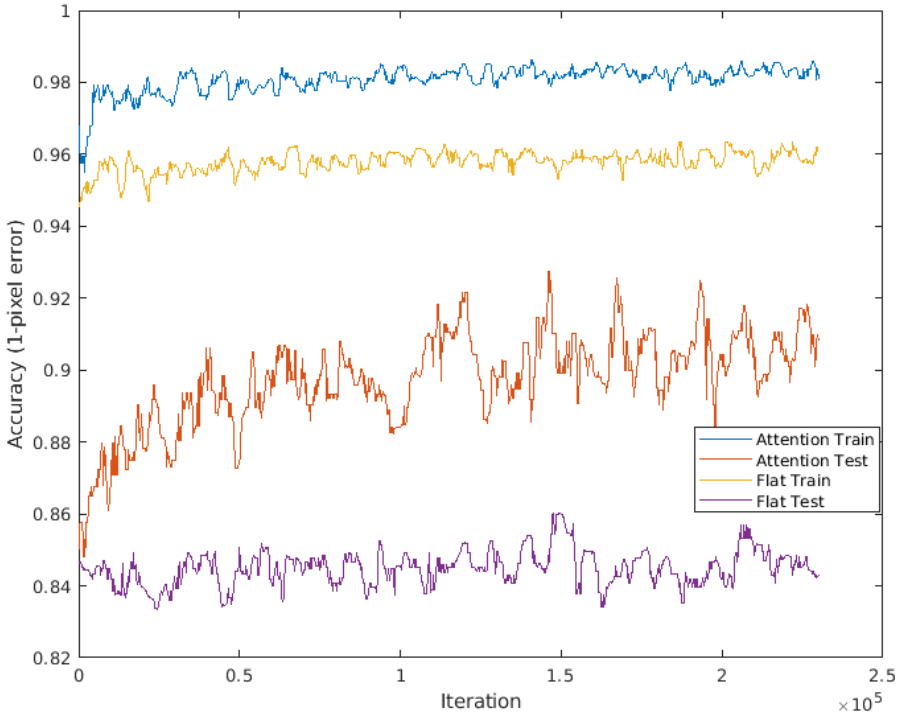


Figure 5.5: Training and validation set accuracy for Flat and Attention neural networks. Training set is Marietta and simulated CCRF, validation set is real CCRF. This shows the significantly better generalization performance of the attention network.

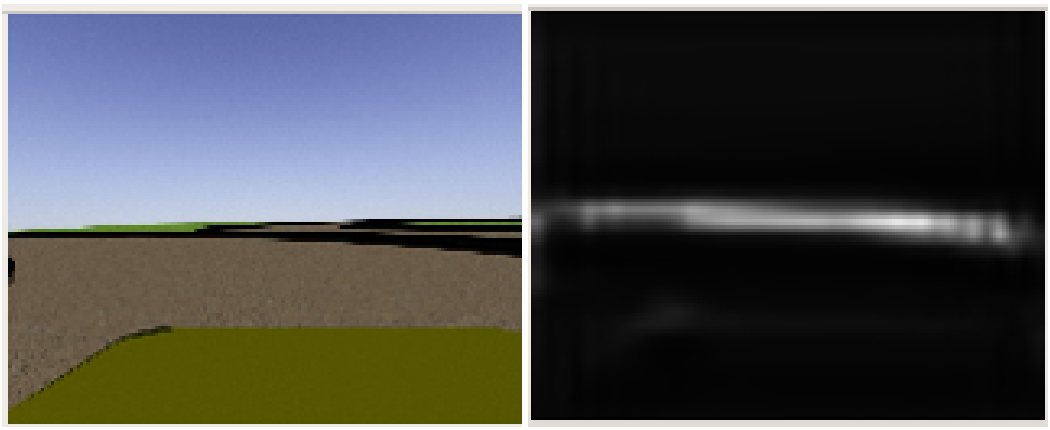


Figure 5.6: Attention example on simulation images. Left shows input image generated from the simulation, right shows attention generated by neural network. Notice that the attention is almost entirely on the barriers, the only feature that will generalize.

The attention network is better able to generalize due to its ability to explicitly ignore distractors and concentrate on the track barriers, which are the most salient features. This can be seen in Fig. 5.5, where the neural network attention concentrates almost exclusively on the track boundaries. However, even with this attention, the neural network is only able to produce a single curve radius when trained on data from the Marietta track, since it has only seen a single corner radius. To combat this limitation and allow generalization from training only on Marietta physical track data to driving at the much more complex CCRF track, we create a very simple simulation of the CCRF track in Gazebo, shown in Fig. 5.4, consisting of barriers and a dirt track surface. While this simulation is very low fidelity, the attention mechanism of the neural network allows effective use of this data to generalize beyond the input distribution.

We first train the attention architecture, shown in Fig. 5.1, on data from the physical Marietta track. This track uses similar looking black borders, and the both tracks are dirt. However, with only this data, the neural network is not able to generalize to driving at the much more complex CCRF track. It is able to negotiate some corners successfully, but is not able to complete this track using the direct driving method. In order to allow the network to generalize, the network is fine tuned with data from the CCRF simulation

(approximately 11 minutes of driving, or 20,000 frames at 30hz). As shown in Fig. 5.6, despite the low fidelity of this simulation environment and difference from the Marietta training distribution, the attention mechanism is able to produce reasonable attention maps. The final network is fine-tuned on a mix of half simulation images and half Marietta images from a network trained only on Marietta data. We find that a longer time horizon produces better results, training providing 64 input images and allowing the network to burn in for 8 images before accumulating error. As shown in Fig 5.5, this method produces a significant decrease in error on a validation set from the physical CCRF track, despite including no actual data from this track. The same training method used with the flat neural network architecture does not reduce error on validation data from the physical CCRF track.

5.5 Experimental Results

When fine tuned from Marietta and simulated CCRF data, the direct driving method is able to successfully complete 3 consecutive laps of the CCRF track. The path taken is shown in left panel of Fig. 5.7. In comparison, on the right panel of Fig. 5.7, the same network structure without attention required intervention from the human operator 3 times to stop a collision, only completing a single lap.

In addition to direct driving, this network was used as input to our particle filter and its error compared with our previous results. As seen in Tab. 5.2, the attention network provides significantly lower position error when used to localize with the particle filter than any other neural network method. Despite never having seen the actual track, the attention network never fails to properly initialize, and has lower average error across these 8 held out runs than any other network trained on the full CCRF dataset. This shows both the generalization power of the attention network and its ability to remove distractors from the image and produce an overall more accurate prediction. This generalization performance can be seen more clearly in Fig. 5.8 where several different training regimes are compared for the recurrent bottleneck and attention architectures.

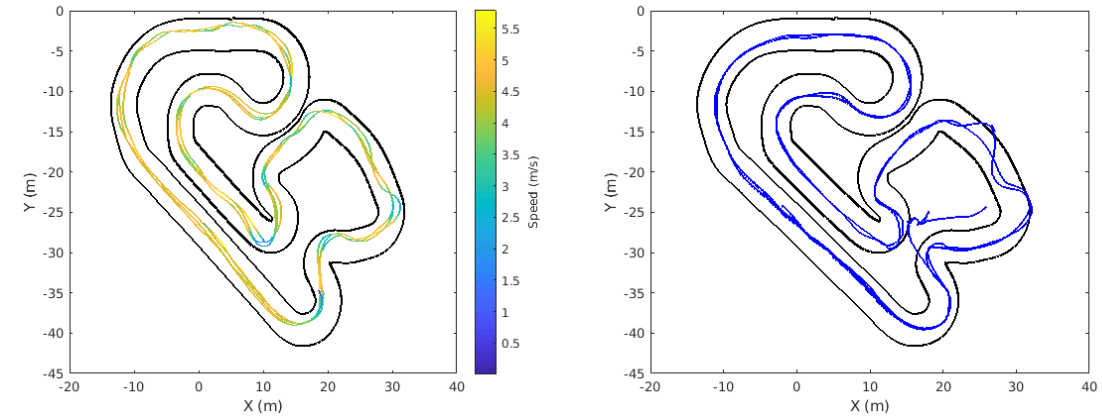


Figure 5.7: (left) Successful 3 lap trial by Direct Driving with network trained only on Marietta and simulated CCRF data. (right) Unsuccessful trial from the same network without attention, with three failures requiring human intervention. Notice some difficulty around the hairpin corner, at coordinate (10,-27), by both networks.

Table 5.2: Particle Filter Position Error for Attention Network With and Without CCRF Training Data

Dataset↓	ED	ED-S	Flat	ED-R	ED-R-S	Attn	Attn-G	ED-R-G
ED	1.21	1.12	1.01	1.19	1.0	0.94	1.01	1.49
ED-S	1.27	1.19	1.02	1.11	1.07	0.91	1.07	1.52
Flat	1.04	0.94	0.92	0.98	0.89	0.87	0.93	1.25
ED-R	1.08	0.99	1.05	1.01	1.05	0.96	1.00	1.35
ED-R-S	1.17	1.01	0.88	1.09	1.09	1.01	1.04	1.34
ED-lodo	1.22	0.91	1.45	1.06	1.05	0.87	0.77	1.57
ED-S-lodo	0.92	0.86	0.87	0.88	0.84	0.79	0.81	1.46
ED-R-S-lodo	0.93	0.96	0.81	1.06	0.80	0.85	0.79	NA
Average	1.11	1.00	1.00	1.05	0.97	0.90	0.93	1.43

ED: Encoder Decoder. R: Recurrent. S: 40x56 output.
Attn: Attention. G: Marietta and Sim CCRF only.
NA: failed to initialize Lowest error

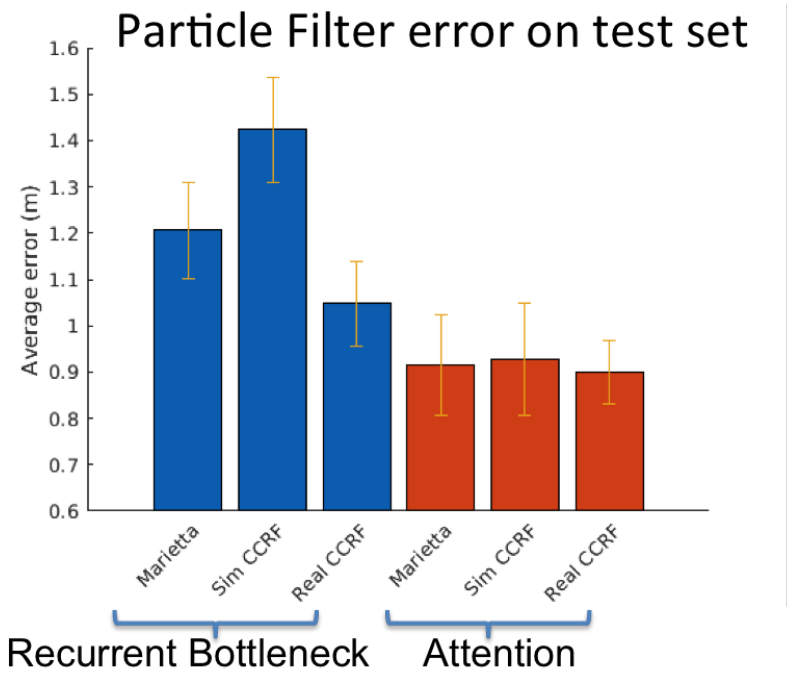


Figure 5.8: Particle filter average error on test set. This shows 3 different training regimes, Marietta data only, simulated CCRF plus Marietta, and real CCRF data for both the recurrent bottleneck architecture and the attention network.

CHAPTER 6
CONCLUSION

In this work, we demonstrate a system capable of repeatable aggressive driving on a complex dirt test track using only monocular vision, IMU, and wheel speed measurements. It combines neural network cost map regression from a monocular camera and model predictive control running in real time on a rugged, high speed autonomous system. Our development and ruggedization of a 1:5 scale autonomous platform enabled this research, and collecting a large corpus of real world aggressive driving data and real world system testing.

We demonstrate and quantify CNN system performance improvement using LSTMs to learn to integrate temporal information. The performance of this system is thoroughly quantified and compared to different architectures and costmap representations. The performance is demonstrated and tested in the real world, using our scaled vehicle testbed and racing facilities.

We demonstrate the ability of a particle filter to integrate this information with IMU and wheel speed sensors and produce a high-rate, high accuracy state estimate. This system is pushed to its performance limits, driving around our track at high speeds while maneuvering aggressively. We show repeatable performance, and use this system to further demonstrate the performance of our complete system.

We demonstrate the ability of our encoder-decoder network to generalize to traversing the track in a direction it has not seen before by performing leave on direction out experiments on both datasets and on the physical system. We develop an attention based neural network that is able to use simulated data to generalize to an unseen environment. This attention network allows a direct comparison of end to end learned attention with human attention on an identical task. The success of the attention mechanism also lends evidence

to the usefulness of our choice of a cost map as an intermediate representation.

These findings extend our knowledge in several ways. The AutoRally platform developed in this work in collaboration with our co-authors helps expand what is possible in the study of aggressive autonomous vehicles and repeatable research. Our investigation into cost map representation and regression from difficult images helps us understand how to more directly use camera images to drive while maintaining understandable intermediate representations and using mature control technology. Our study of recurrence and attention demonstrate promising avenues for neural network research, and help show the real-world performance of many different architectures.

6.1 Future Work

As with any real-world system, we find that details significantly improve the performance of the system. Reducing system lag by A) propagating state forward at 200 Hz using the IMU measurements, B) threading software systems where possible to reduce latency and C) forward propagating the model predictive control outputs using state feedback gains all help to push the system to its limit. When training our neural network, careful curation of the dataset, model architecture tuning, and hyper-parameter tuning all improve final system performance. Careful system identification dataset collection improves the accuracy and predictive power of the vehicle dynamics model. Finally, having a system that is robust and able to be repeatedly pushed to the limits of handling (and beyond) is crucial to iterative design process needed to push the system to the limits of handling and grip.

The attention mechanism introduced in this dissertation is an excellent candidate for further study. We have shown the beginnings of how it might be used in sim to real transfer, and how it can greatly improve generalization to unseen environments. Future avenues to explore include full, simulation only training and how much fidelity is required in the simulation to operate in the real world. Additionally, there is room for improvement in the architecture to allow for longer temporal training, and learning of long term dependencies.

This attention mechanism has great potential to shine in obstacle detection regimes. Because the attention is generated in a fully convolutional setting, it should have a great deal of position covariance. This should greatly help obstacle detection and recognition algorithms remove clutter and generalize to new environments. There is potential for the temporal nature of the training to help detect novel obstacles as well.

In conclusion, there are many interesting avenues left open for exploration. We have demonstrated excellent performance in the real world with a combination of emerging technologies and new development. We hope that this work can enlighten future researchers and engineers.

Appendices

APPENDIX A

NETWORK ERROR VS POSITION GRAPHS

The particle filter position error vs track position graphs for the attention neural network trained on Marietta and simulated CCRF data are shown in Figs. A.1 and A.2. The same graphs for the attention neural network trained with Marietta and real CCRF data is shwon in Figs. A.3 and A.4.

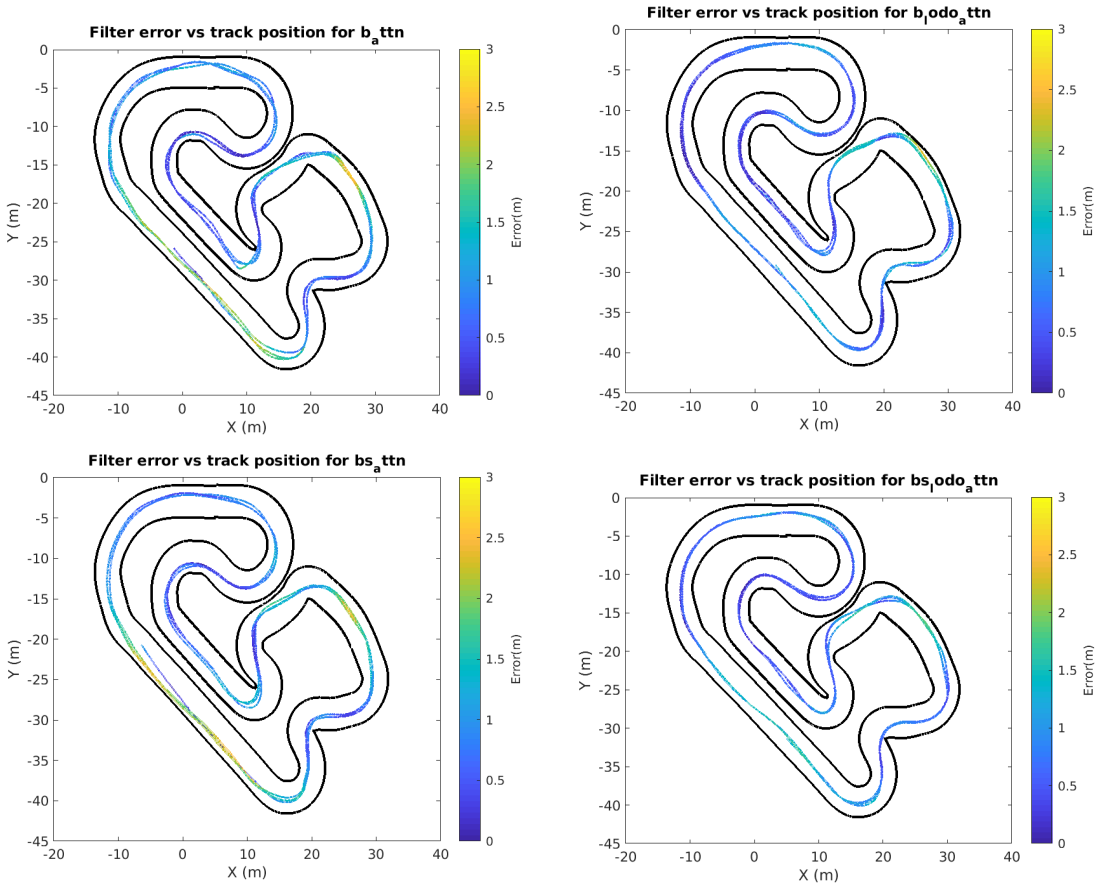


Figure A.1: Particle Filter position estimation error.

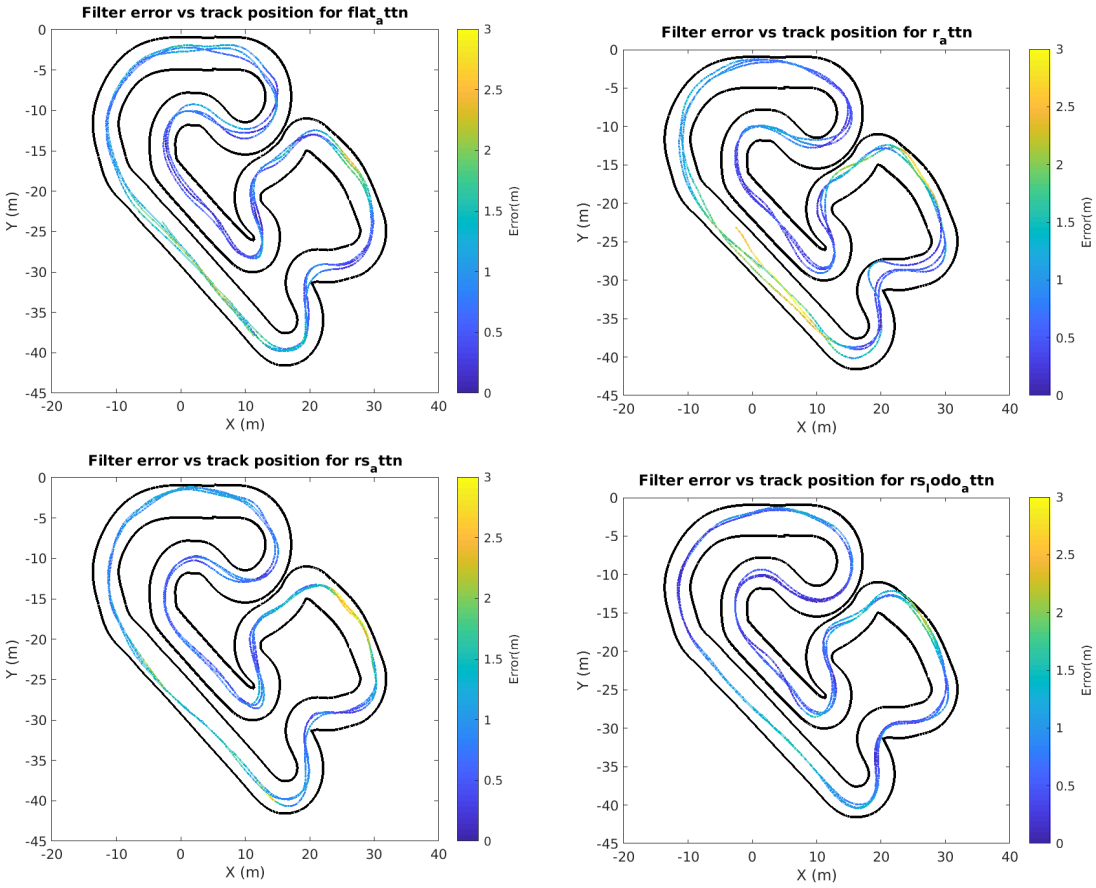


Figure A.2: Particle Filter position estimation error.

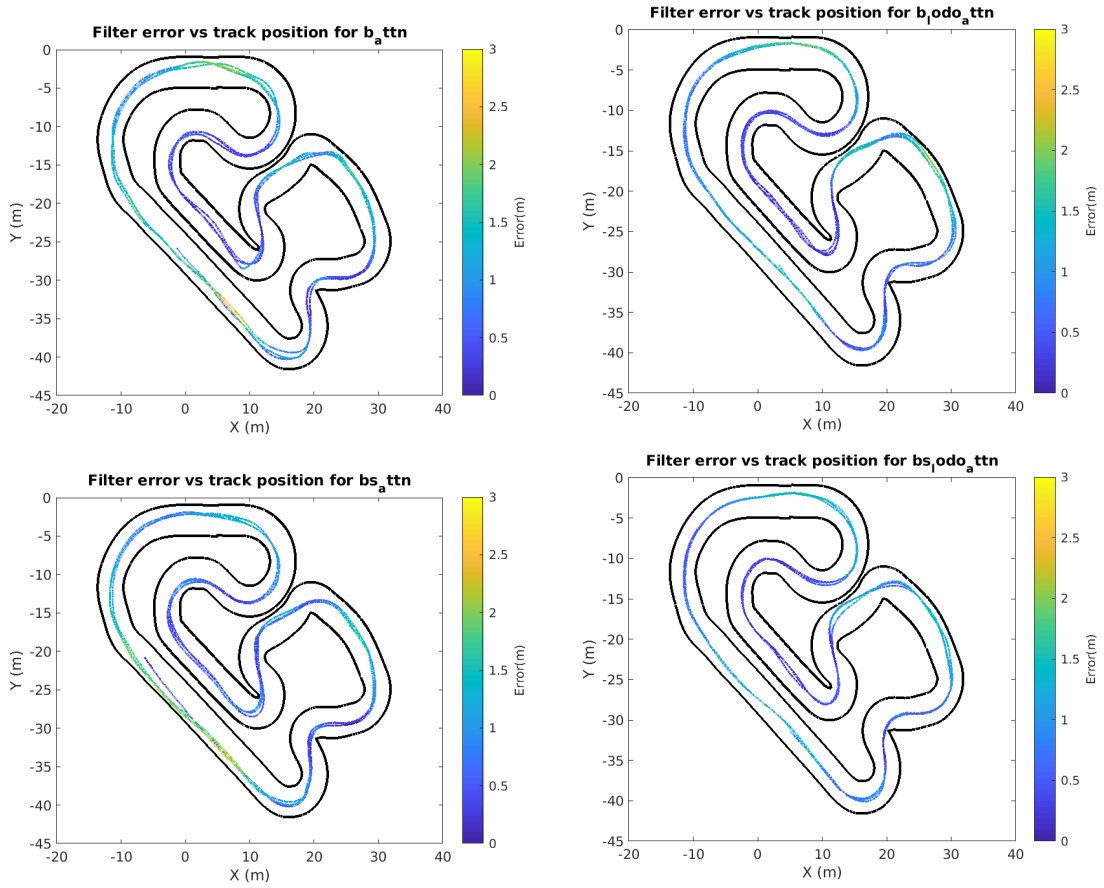


Figure A.3: Particle Filter position estimation error.

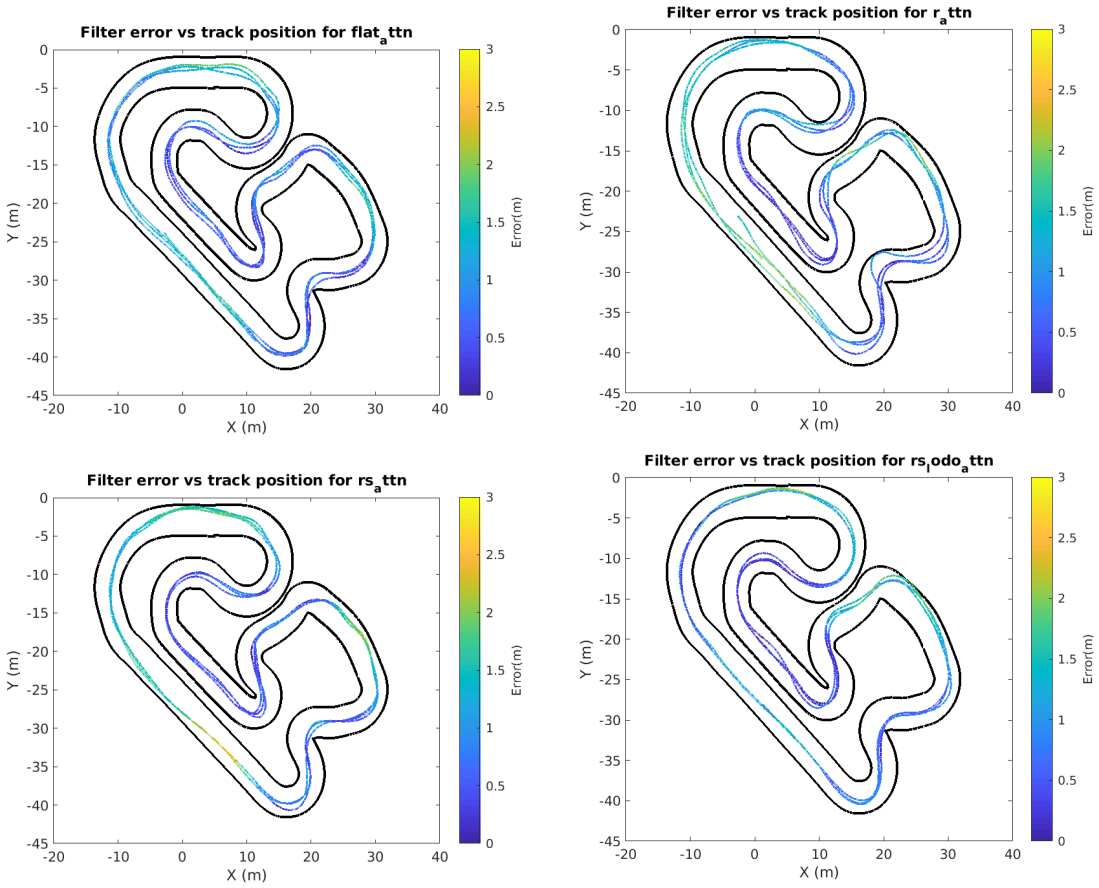


Figure A.4: Particle Filter position estimation error.

APPENDIX B

ATTENTION COMPARISONS

This chapter contains further comparisons between the attention generated by the neural network and human attention on the track. These figures show average gaze locations for individual participants along with neural network gaze at a single central location and a map of human gaze locations.

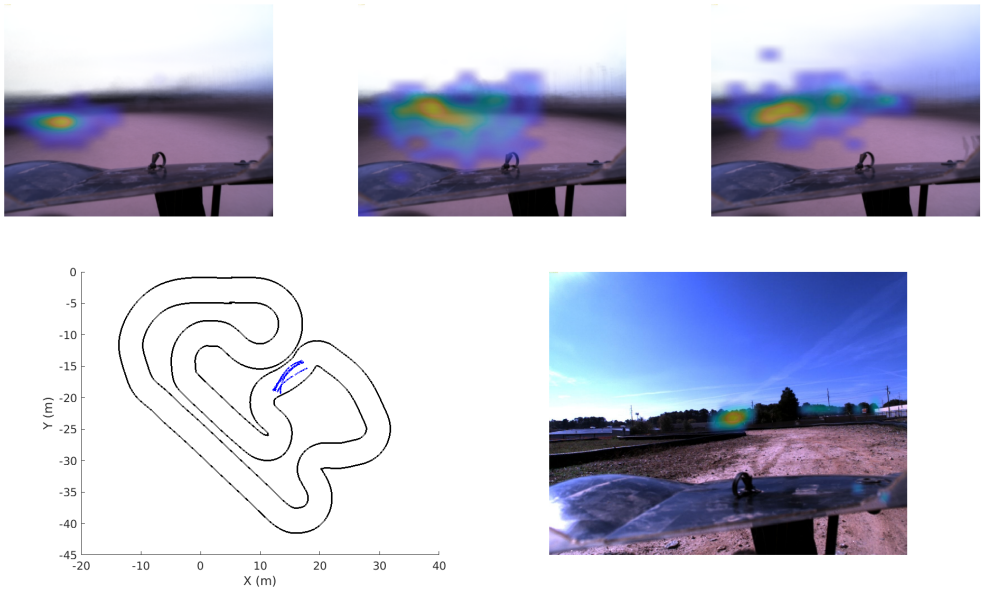


Figure B.1: Human and CNN attention location comparison. Top row: Human gaze heatmaps. Gaze locations from participants 1, 2, and 3 are averaged over a local track location leading up to a corner. Bottom row: Location of averaged human gaze and CNN attention heat map from same track location.

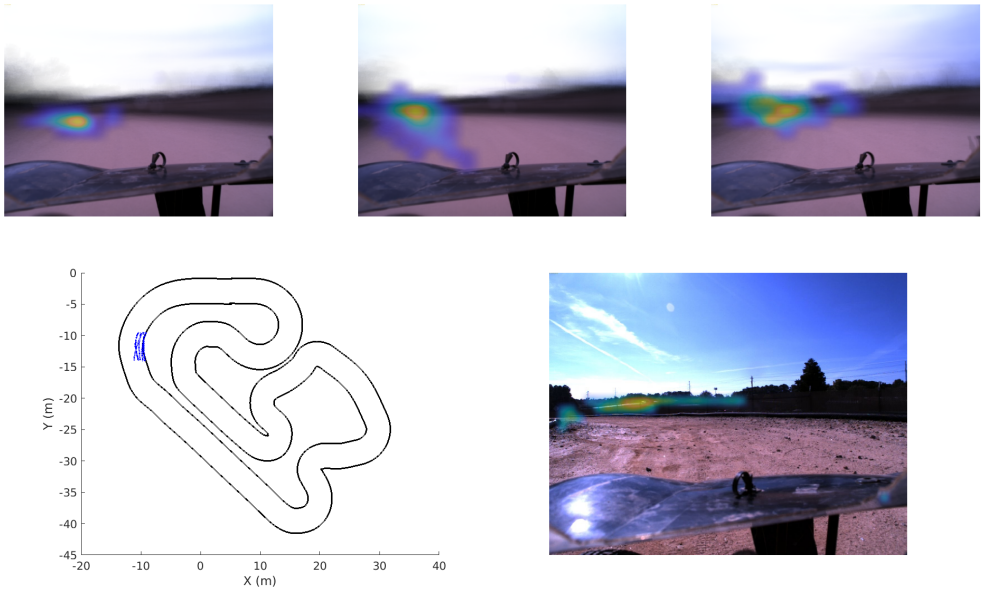


Figure B.2: Human and CNN attention location comparison. Top row: Human gaze heatmaps. Gaze locations from participants 1, 2, and 3 are averaged over a local track location leading up to a corner. Bottom row: Location of averaged human gaze and CNN attention heat map from same track location.

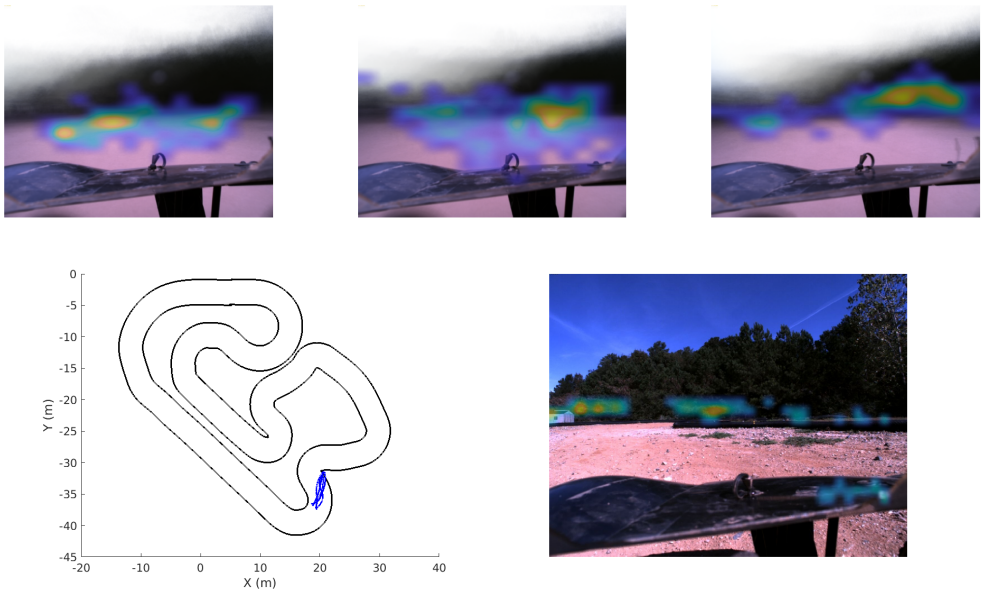


Figure B.3: Human and CNN attention location comparison. Top row: Human gaze heatmaps. Gaze locations from participants 1, 2, and 3 are averaged over a local track location leading up to a corner. Bottom row: Location of averaged human gaze and CNN attention heat map from same track location.

REFERENCES

[1] M. Montemerlo *et al.*, "Junior: The stanford entry in the urban challenge", *Journal of field robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[2] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge", *Journal of field robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[3] Urmson *et al.*, "Tartan racing: A multi-modal approach to the darpa urban challenge", 2007.

[4] D. I. Katzourakis, I. Papaefstathiou, and M. G. Lagoudakis, "An open-source scaled automobile platform for fault-tolerant electronic stability control", *Instrumentation and measurement, ieee transactions on*, vol. 59, no. 9, pp. 2303–2314, 2010.

[5] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam", in *European conference on computer vision*, Springer, 2014, pp. 834–849.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system", *Ieee transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs", *Arxiv preprint arxiv:1412.7062*, 2014.

[8] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving", in *Proceedings of the ieee international conference on computer vision*, 2015, pp. 2722–2730.

[9] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision", *International journal of computer vision*, vol. 1, no. 4, pp. 333–356, 1988.

[10] L. Itti and C. Koch, "Computational modelling of visual attention", *Nature reviews neuroscience*, vol. 2, no. 3, pp. 194–203, 2001.

[11] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention", in *International conference on machine learning*, 2015, pp. 2048–2057.

[12] Donkey Car. Available online at `http://www.donkeycar.com/` (last accessed November, 2017).

[13] J Gonzales, F Zhang, K Li, and F Borrelli, "Autonomous drifting with onboard sensors", in *Advanced vehicle control: Proceedings of the 13th international symposium on advanced vehicle control (avec'16), september 13-16, 2016, munich, germany*, CRC Press, 2016, p. 133.

[14] F1/10. (2016). F1/10 autonomous racing competition, (visited on 02/28/2017).

[15] Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot (RACECAR). Available online at `https://mit-racecar.github.io/` (last accessed November, 2017).

[16] N. Keivan and G. Sibley, "Realtime simulation-in-the-loop control for agile ground vehicles", in *Conference towards autonomous robotic systems*, Springer, 2013, pp. 276–287.

[17] J. L. Jakobsen, *Autonomous drifting of a 1:5 scale model car*, 2011.

[18] S. Song, "Towards autonomous driving at the limit of friction", 2015.

[19] M. Cutler, T. J. Walsh, and J. P. How, "Reinforcement learning with multi-fidelity simulators", in *Robotics and automation (icra), 2014 ieee international conference on*, IEEE, 2014, pp. 3888–3895.

[20] W. E. Travis, R. J. Whitehead, D. M. Bevly, and G. T. Flowers, "Using scaled vehicles to investigate the influence of various properties on rollover propensity", in *American control conference, 2004.*, IEEE, vol. 4, 2004, pp. 3381–3386.

[21] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles", *Ieee transactions on intelligent vehicles*, vol. 1, pp. 33–55, 2016.

[22] Thrun *et al.*, "Stanley: The robot that won the darpa grand challenge", *Journal of field robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[23] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot", vol. 229, pp. 171–253, 1998.

[24] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, *et al.*, "Odin: Team victortango's entry in the darpa urban challenge", *Journal of field robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[25] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, *et al.*, "A perception-driven autonomous urban vehicle", *Journal of field robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[26] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving", vol. 1, 2008.

[27] T Fraichard and R Mermond, "Path planning with uncertainty for car-like robots", in *Ieee international conference on robotics and automation (icra)*, 1998, pp. 27–32.

[28] R. Pepy and A. Lambert, "Safe path planning in an uncertain-configuration space using rrt", in *Ieee/rsj internation conference on intelligent robots and systems*, IEEE, 2006, pp. 5376–5381.

[29] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving", *Ieee transactions on control systems technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[30] R. Pepy, A. Lambert, and H. Mounier, "Path planning using a dynamic vehicle model", in *Information and communication technologies (ictta)*, vol. 1, 2006, pp. 781–786.

[31] E. Velenis, P. Tsiotras, and J. Lu, "Modeling aggressive maneuvers on loose surfaces: The cases of trail-braking and pendulum-turn", in *Control conference (ecc), 2007 european*, IEEE, 2007, pp. 1233–1240.

[32] M. Gerdts, S. Karrenberg, B. Müller-Beßler, and G. Stock, "Generating locally optimal trajectories for an automatically driven car", *Optimization and engineering*, vol. 10, no. 4, pp. 439–463, 2009.

[33] P. Tsiotras and R. S. Diaz, "Real-time near-optimal feedback control of aggressive vehicle maneuvers", in *Optimization and optimal control in automotive systems*, Springer, 2014, pp. 109–129.

[34] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis", *Ieee transactions on pattern analysis & machine intelligence*, no. 11, pp. 1254–1259, 1998.

[35] V. Mnih, N. Heess, A. Graves, *et al.*, "Recurrent models of visual attention", in *Advances in neural information processing systems*, 2014, pp. 2204–2212.

[36] J. Aloimonos, "Purposive and qualitative active vision", in *Pattern recognition, 1990. proceedings., 10th international conference on*, IEEE, vol. 1, 1990, pp. 346–360.

[37] R. Bajcsy, "Active perception", *Proceedings of the ieee*, vol. 76, no. 8, pp. 966–1005, 1988.

[38] R. Bajcsy and P. K. Allen, *Converging disparate sensory data*. University of Pennsylvania, Department of Computer and Information Science, 1984.

[39] M. Chli and A. J. Davison, "Active matching", in *Computer vision–eccv 2008*, Springer, 2008, pp. 72–85.

[40] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision", *Ieee transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 865–880, 2002.

[41] A. Kelly and A. Stentz, "Rough terrain autonomous mobilitypart 2: An active vision, predictive control approach", *Autonomous robots*, vol. 5, no. 2, pp. 163–198, 1998.

[42] A. L. Abbott and N. Ahuja, "Surface reconstruction by dynamic integration of focus, camera vergence, and stereo", in *Computer vision., second international conference on*, IEEE, 1988, pp. 532–543.

[43] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson, "Contextual guidance of eye movements and attention in real-world scenes: The role of global features in object search.", *Psychological review*, vol. 113, no. 4, p. 766, 2006.

[44] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency", in *Advances in neural information processing systems*, 2007, pp. 545–552.

[45] X. Hou, J. Harel, and C. Koch, "Image signature: Highlighting sparse salient regions", *Ieee transactions on pattern analysis and machine intelligence*, vol. 34, no. 1, pp. 194–201, 2012.

[46] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach", in *Computer vision and pattern recognition, 2007. cvpr'07. ieee conference on*, IEEE, 2007, pp. 1–8.

[47] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu, "Global contrast based salient region detection", in *Computer vision and pattern recognition (cvpr), 2011 ieee conference on*, IEEE, 2011, pp. 409–416.

[48] Y. Li, A. Fathi, and J. M. Rehg, "Learning to predict gaze in egocentric video", in *The ieee international conference on computer vision (iccv)*, 2013.

[49] M. F. Land and D. N. Lee, "Where we look when we steer", *Nature*, vol. 369, no. 6483, pp. 742–744, 1994.

[50] M. F. Land and B. W. Tatler, "Steering with the head: The visual strategy of a racing driver", *Current biology*, vol. 11, no. 15, pp. 1215 –1220, 2001.

[51] D. D. Salvucci and R. Gray, "A two-point visual control model of steering", *Perception*, vol. 33, no. 10, pp. 1233–1248, 2004.

[52] K. D. Robertshaw and R. M. Wilkie, "Does gaze influence steering around a bend?", *Journal of vision*, vol. 8, no. 4, pp. 18–18, 2008.

[53] R. Wilkie and J. Wann, "Controlling steering and judging heading: Retinal flow, visual direction, and extraretinal information.", *Journal of experimental psychology: Human perception and performance*, vol. 29, no. 2, p. 363, 2003.

[54] R. M. Wilkie, J. P. Wann, and R. S. Allison, "Active gaze, visual look-ahead, and locomotor control", *Journal of experimental psychology: Human perception and performance*, vol. 34, no. 5, pp. 1150–1164, 2008.

[55] A. Borji, D. N. Sihite, and L. Itti, "Probabilistic learning of task-specific visual attention", in *Computer vision and pattern recognition (cvpr), 2012 ieee conference on*, IEEE, 2012, pp. 470–477.

[56] L. Johnson, B. Sullivan, M. Hayhoe, and D. Ballard, "Predicting human visuomotor behaviour in a driving task", *Philosophical transactions of the royal society of london b: Biological sciences*, vol. 369, no. 1636, p. 20 130 044, 2014.

[57] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine", in *Advances in neural information processing systems*, 2010, pp. 1243–1251.

[58] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention", *Arxiv preprint arxiv:1412.7755*, 2014.

[59] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation", *Arxiv preprint arxiv:1502.04623*, 2015.

[60] A. Ablavatski, S. Lu, and J. Cai, "Enriched deep recurrent visual attention model for multiple object recognition", in *Applications of computer vision (wacv), 2017 ieee winter conference on*, IEEE, 2017, pp. 971–978.

[61] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning", *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[62] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Mller-Bessler, and B. Huhnke, "Up to the limits: Au-

tonomous audi tts", in *2012 ieee intelligent vehicles symposium*, 2012, pp. 541–547.

[63] N. Keivan and G. Sibley, "Realtime simulation-in-the-loop control for agile ground vehicles", in *Towards autonomous robotic systems: 14th annual conference, taros 2013, oxford, uk, august 28–30, 2013, revised selected papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 276–287, ISBN: 978-3-662-43645-5.

[64] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control", *Ieee international conference on robotics and automation*, 2016 (submitted).

[65] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.", in *Robotics: Science and systems*, vol. 2, 2014.

[66] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking", in *Mixed and augmented reality (ismar), 2011 10th ieee international symposium on*, IEEE, 2011, pp. 127–136.

[67] C. Beall and F. Dellaert, "Appearance-based localization across seasons in a metric map", *6th ppniv, chicago, usa*, 2014.

[68] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization", *Proceedings of the international conference on robotics and automation (ICRA)*, 2016.

[69] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars", *Arxiv preprint arxiv:1604.07316*, 2016.

[70] D. A. Pomerleau, "Alvinn, an autonomous land vehicle in a neural network", Carnegie Mellon University, Computer Science Department, Tech. Rep., 1989.

[71] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies", *Journal of machine learning research*, vol. 17, no. 39, pp. 1–40, 2016.

[72] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning", in *Proceedings of robotics: Science and systems*, Pittsburgh, Pennsylvania, 2018.

[73] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search", in *2016 ieee international conference on robotics and automation (icra)*, 2016, pp. 528–535.

[74] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control", in *Robotics: Science and systems*, 2015.

[75] N. Wahlstrm, T. B. Schn, and M. P. Deisenroth, "Learning deep dynamical models from image pixels", *Ifac-papersonline*, vol. 48, no. 28, pp. 1059 –1064, 2015, 17th IFAC Symposium on System Identification SYSID 2015.

[76] C. Schenck and D. Fox, "Visual closed-loop control for pouring liquids", May 2017, pp. 2629–2636.

[77] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving", *Journal of field robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[78] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding", in *Proc. of the ieee conference on computer vision and pattern recognition (cvpr)*, 2016.

[79] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, "Semantic segmentation using regions and parts", in *Computer vision and pattern recognition (cvpr), 2012 ieee conference on*, IEEE, 2012, pp. 3378–3385.

[80] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[81] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions", in *Iclr*, 2016.

[82] C. Richter, J. Ware, and N. Roy, "High-speed autonomous navigation of unknown environments using learned probabilities of collision", in *Robotics and automation (icra), 2014 ieee international conference on*, IEEE, 2014, pp. 6114–6121.

[83] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection", 2017.

[84] J. Hays and A. A. Efros, "Im2gps: Estimating geographic information from a single image", in *Computer vision and pattern recognition, 2008. cvpr 2008. ieee conference on*, IEEE, 2008, pp. 1–8.

[85] I. Miller and M. Campbell, "Particle filtering for map-aided localization in sparse gps environments", in *2008 ieee international conference on robotics and automation*, 2008, pp. 1834–1841.

[86] C. Rose, J. Britt, J. Allen, and D. Bevly, "An integrated vehicle navigation system utilizing lane-detection and lateral position estimation systems in difficult environments for gps", *Ieee transactions on intelligent transportation systems*, vol. 15, no. 6, pp. 2615–2629, 2014.

[87] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsiotras, and J. M. Rehg, *Autorally an open platform for aggressive autonomous driving*, 2018. eprint: arXiv:1806.00678.

[88] B. Goldfain. (2017). Autorally platform instructions, (visited on 05/22/2017).

[89] G. Williams, A. Aldrich, and E. Theodorou, "Model Predictive Path Integral Control using Covariance Variable Importance Sampling", *Arxiv e-prints*, Sep. 2015. arXiv: 1509.01149.

[90] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning", in *International conference on robotics and automation (icra)*, 2017.

[91] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation", *Journal of guidance, control, and dynamics*, pp. 1–14, 2017.

[92] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "Isam2: Incremental smoothing and mapping using the bayes tree", *International journal of robotics research*, vol. 31, no. 2, SI, pp. 216–235, 2012.

[93] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing", *The international journal of robotics research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[94] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation", in *Proceedings of robotics: Science and systems*, Rome, Italy, 2015.

[95] N. Keivan and G. Sibley, "Realtime simulation-in-the-loop control for agile ground vehicles", in *Towards autonomous robotic systems*, Springer, 2014, pp. 276–287.

[96] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *Corr*, vol. abs/1412.6980, 2014.

[97] M. Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015.

[98] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation", in *The ieee international conference on computer vision (iccv)*, 2015.

[99] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting", in *Advances in neural information processing systems*, 2015, pp. 802–810.

[100] D. D. Salvucci and R. Gray, "A two-point visual control model of steering", *Perception*, vol. 33, no. 10, pp. 1233–1248, 2004.

VITA

Paul Drews received the B.S. in Electrical Engineering and Computer Engineering from the Missouri University of Science and Technology in 2008. He worked for several years on autonomous aerial vehicle technology, and supporting payload integration and flight testing. He then decided to continue his education as a Robotics PhD student in the School of Electrical and Computer Engineering at Georgia Institute of Technology in order to work at the cutting edge of robotics and automation. He is an avid supporter of the outdoors, spending much of his free time running, cycling, climbing and kayaking with his wife.